Informe de Proyecto de Grado

Instituto de Ingeniería Eléctrica − Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
Abril 2012

Integrantes: Santiago Barreira, Leticia Grassi, Johnny Silvera.
Tutor: Ing. Eduardo Cota

Resumen

Este trabajo se centra en sintetizar e introducir los conceptos de la extensión de seguridad del Sistema de Nombres de Dominio −DNS por sus siglas en Inglés−, las cuales se denominan DNSSec −DNS Security−. Estas extensiones tienen como objetivo garantizar la autenticidad e integridad de datos en las interacciones entre los servidores autoritativos de nombre y los servidores recursivos o clientes. El sistema hace uso de claves públicas que se distribuyen en toda la jerarquía del árbol de nombres, las cuales se utilizan para validar las firmas digitales que autentican los datos. En esta documentación se desarrollan todos los procedimientos involucrados en DNSSec, resaltando entre estas la construcción de la cadena de confianza por medio de delegaciones seguras, la administración de claves y la validación de datos existentes y no existentes.

El proyecto también incluyó la elaboración de una maqueta experimental, para reproducir el comportamiento de los servidores de nombre autoritativos y recursivos frente a los nuevos registros y procesos descritos en DNSSec. Se abordó el problema del incremento de recursos que la implantación de DNSSec trae asociado, comparando la performance de los servidores sirviendo zonas sin firmar y firmadas. Se encontró que el ancho de banda en un servidor autoritativo sirviendo zonas firmadas se incrementa un 50% con respecto al ancho de banda necesario para servir una zona sin firmar. Por otro lado, el servidor recursivo se verá mayormente afectado por la carga sobre su CPU al realizar las validaciones tanto de firmas digitales, incluyendo la construcción de la cadena de confianza, como de la verificación de los registros NSEC3 necesarios para la validación de respuestas de no existencia.

Agradecimientos

Queremos dejar expresado nuestro agradecimiento a Mónica Soliño, Luis Castillo y Sergio Ramírez del Servicio Central de Informática de la Universidad, quienes nos brindaron información y material sobre sus servidores DNS los cuales fueron muy valiosos para el desarrollo del proyecto. También queremos agradecer a Carlos Martínez-Cagnazzo quien nos invitó a participar de un evento de LACNIC, en el cual pudimos presenciar conferencias de DNSSec dictadas por personas directamente vinculadas al despliegue y al desarrollo de DNSSec. Por último, un agradecimiento especial a nuestro tutor, quien con su constante apoyo, dedicación y guía a lo largo de todo este tiempo fue posible llevar a término este proyecto.

Tabla de Contenido

Introducción

El objetivo de este proyecto de grado fue introducirse en las nuevas extensiones de seguridad para el Sistema de Nombre de Dominio, DNSSec.

En primer lugar se presenta un estudio del DNS, e el cual se desarrolla en el capítulo inicial, donde se describen sus principales conceptos, características y debilidades. Se introducen las principales operaciones del DNS como preguntas recursivas e iterativas, las cuales son relevantes para la comprensión del DNS. Se describen varios ataques contra el DNS los cuales DNSSec minimiza, como pueden ser el Man in the Middle y el Spoofing, y otros los cuales la implementación de DNSSec los puede hacer más grave como por ejemplo ataques de denegación de servicio.

Para introducir el DNSSec se mencionan temas actuales que involucran tanto a DNS como a DNSSec. Luego se definen los principales conceptos de DNSSec, nuevas consideraciones que se deben tener en cuenta al implementar estas extensiones en los servidores recursivos y autoritativos, y nuevas consideración de seguridad. También se describen los nuevos Registros de Recursos que hacen de DNSSec una extensión robusta.

Se realiza hincapié en los problemas que introdujo DNSSec al introducir NSEC y como se logro minimizarlas al implementar el nuevo registro NSEC3. Se desarrolla una importante descripción de estos dos registros, implementación y parámetros que hacen de estos una de las partes más complejas pero interesantes de DNSSec.

En base a todo lo anterior se realizó una maqueta donde se simuló la interacción entre servidores autoritativos y recursivos, donde se estudió la performance de estos al implementar DNSSec. Verificándose incrementos en el uso de ancho de banda y CPU comparado con el caso en que no se utiliza DNSSec. También se obtuvieron resultados interesantes de la comparación entre los dos mecanismos para la validación de respuestas de no existencia de un registro, NSEC y NSEC3.

1. Sistema de Nombres de Dominio

1.1 - Breve Historia

En los años 70 el ímpetu por el desarrollo en internet iba en crecimiento así como también el del sistema de dominios. Al inicio, el mapeo de direcciones a nombres de host era gestionado por el Network Information Center −NIC−, en un único fichero HOSTS.txt, el cual era distribuido a todos los host mediante FTP. Esto llevó a un consumo de ancho de banda total en la red, proporcional al cuadrado del número de los host de la red. Si bien en su momento esto no presentó un inconveniente, el explosivo crecimiento de Internet dejaría en evidencia que se precisaría un nuevo sistema de nombres.

Surgió entonces la necesidad de tener un servicio de nombres de propósitos más generales. Se planteó la idea de un espacio de nombres jerárquico, el uso del "." como carácter de unión entre los niveles jerárquicos y un diseño basado en una base de datos distribuida y de recursos generalizados. Nace así el DNS en 1983, cuando Paul Mockapetris publicó las RFCs 882 [1] y 883 [2], definiendo lo que hoy en día ha evolucionado hacia el DNS moderno. Estos RFCs han quedado obsoletos por la publicación en 1987 de los RFCs 1034 [3] y 1035 [4], los cuales a su vez fueron actualizados por otras RFCs −en particular las de DNSSec−.

El DNS permitió el mapeo de direcciones IP en un nombre fácil de recordar para los usuarios. Rápidamente los servidores de nombre de dominio crecieron en relevancia, convirtiéndose en sistemas críticos para cualquier tipo de red, especialmente Internet.

Desde su inicio [1] el DNS sostuvo tres requisitos importantes que debía cumplir, los cuales le permitió crecer y adaptarse al rápido crecimiento de Internet.

1. Jerarquía de nombres.
2. Repartir la carga entre los servidores de nombres de dominio.
3. Delegar la administración de diferentes servidores.

1.2 - Elementos del DNS

El DNS tiene tres componentes principales:

- *El espacio de Nombres de Dominio y Registros de Recursos*: El espacio de nombres se organiza en una estructura de árbol, cada nodo y hoja de este árbol define un conjunto de información. Esta información puede ser accedida por medio de consultas que indican el nombre de dominio −nodo− y registro de recurso −información− que desean obtener.
- *Servidores de Nombres*: Se denomina Servidor de Nombres, al programa que administra −almacena, modifica, etc.− alguna parte −zona− del espacio de nombres y a sus registros de recursos −RR−. Estos servidores tienen la característica de responder de forma autoritativa a consultas referidas a nombres de dominio dentro de la zona que este administra. Las respuestas autoritativas se identifican por tener el bit AA −Respuesta autoritativa por sus siglas en Inglés− en el encabezado del paquete DNS. Este bit estará seteado si la respuesta proviene directamente de un servidor de nombres autoritativo o si proviene de un servidor recursivo que no tenia dicha información en. Los operadores de estos servidores autoritativos tienen la responsabilidad de mantener actualizados los dominios y sus registros dentro de la zona.
- *Servidor Recursivo*: También conocido como resolver, son programas responsables de resolver consultas por nombres de dominios y sus recursos. Estos servidores recorren jerárquicamente el árbol de nombres, comenzando desde la raíz –si es que no tienen información más especifica almacenada en cache– extrayendo la información que precisan de los servidores de nombres. La información que obtienen es almacenda en cache, de forma de poder responder la próxima consulta más rápidamente −y consumiendo menos recursos de la red−.
- *Stub-resolver*: Son implementaciones limitadas de servidores recursivos presentes en los clientes. Su función es reenviar las consultas DNS a servidores recursivos, ya que los stub-resolver no pueden seguir las referencias enviadas por otros servidores autoritativos.

1.3 - El Espacio de Nombres

1.3.1 - Especificaciones y terminologías

El espacio de nombres de dominio está estructurado en forma de árbol. Cada nodo y hoja en el árbol corresponde a un nombre de dominio y tiene asociado un conjunto de recursos. El sistema de dominio no distingue entre el uso de los nodos interiores y hojas, por lo que en este documento utilizaremos el término "nodo" para referirnos a ambos.

Cada nodo tiene una etiqueta con un tamaño que varía entre 0 y 63 bytes, siendo el tamaño total del nombre de dominio menor a 255 bytes. Los nodos hermanos no deben tener la misma etiqueta, mientras que la misma etiqueta puede ser utilizada por nodos que no son hermanos. Existen varias etiquetas reservadas [5] en diferentes niveles del árbol. La más importante es la etiqueta vacía −NULL−, reservada para representar a la raíz. Otra etiqueta, no tan importante, es la example, la cual está reservada bajo el nodo com, org y net, con el propósito de ser utilizadas en documentación y para ilustrar ejemplos.

El nombre de dominio que representa un nodo del espacio de nombres, está compuesto por las etiquetas recorridas desde el mismo nodo hasta la raíz del árbol. Por convención, las etiquetas que componen un nombre de dominio se leen de izquierda a derecha, desde la más específica −la más baja, lejos de la raíz− a la menos específica −la más alta, cerca de la raíz−.

También fue establecido que los nombres de dominio fueran insensibles a las mayúsculas y minúsculas, lo que significa que una búsqueda por FING.edu.uy es indistinguible de una búsqueda por fing.edu.uy, sin importar de qué forma haya sido almacenado el dominio en el servidor de nombres.

Los nombres de dominio se pueden clasificar en absolutos o relativos, también denominados FQDN −Fully Qualified Domain Name− o no FQDN respectivamente. Un nombre completo −absoluto− debería terminar con la etiqueta raíz −la etiqueta nula−. Esta propiedad sirve para distinguir entre estos casos:

- Un string de caracteres que representa un nombre de dominio completo. Por ejemplo, "iie.fing.edu.uy."
- Un string de caracteres que representa el comienzo de etiquetas de un nombre de dominio completo. Este debería ser completado por algún software con información del dominio local −relativo−. Por ejemplo, "iie" dentro del dominio fing.edu.uy.

Los nombres relativos pertenecen a un origen bien conocido o se pueden encontrar en una lista local de dominios. Los nombres relativos aparecen con mayor frecuencia en la interfaz de usuario, donde su representación varía entre implementaciones. La interpretación más común utiliza la raíz "." como el origen, por eso un nombre relativo con varias etiquetas suele omitir el punto raíz para ahorrar un carácter.

1.3.2 - Jerarquía de nombres

Cada nivel del árbol de nombres representa un nivel jerárquico. En el nivel más alto está la raíz, seguido de los dominios de nivel superior −TLD, por las siglas en Inglés de Top Level Domain−, después los dominios de segundo nivel −SLD, Second Level Domain− y finalmente otros niveles; todos son separados por un punto. Los TLD se dividen en dos tipos básicos:

- Dominios de nivel superior genéricos –gTLD, Generic Top Levels Domains–: .com, .edu, .net, .org, .mil.
- Dominios de nivel superior de códigos de países –ccTLD, Country Code Top Levels Domains– que se identifican con una secuencia de dos caracteres según la ISO 3166-1: .uy, .us, .ar, .uk.

Los ccTLD son delegados generalmente a organizaciones gubernamentales, cuya relación con ICANN –Internet Corporation for Assigned Numbers and Names– es principalmente consultiva más que contractual, debido a razones culturales y de soberanía de cada país. La Internet Assigned Numbers Authority –IANA– es la encargada de mantener actualizada la lista de los administradores de cada ccTLD.

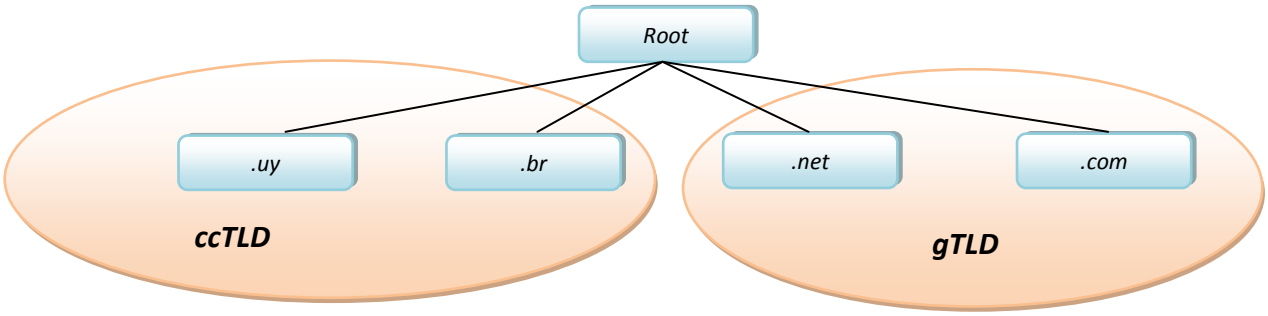


Figura 1.1: ccTLD y gTLD

1.3.3 - Autoridad

A cada nodo del árbol DNS se le es asignado una autoridad, una organización o persona, responsable de la gestión y operación del nodo. Dicha organización, administra su nodo de forma autoritativa, por lo tanto puede definir todos los recursos bajo dicho nodo, así como también delegar la autoridad de nodos inferiores a él a otras organizaciones.

La autoridad de la raíz –root domain– fue delegada a la Internet Corporation for Assigned Numbers and Names –*ICAN, www.icann.org*–. En 1998, esta organización sin fines de lucro, asumió esta responsabilidad en el Departamento de Comercio de Estados Unidos. Las gTLD están autoritativamente administradas por ICANN y la autoridad de las ccTLD es delegada a cada país, los que eligen como administrarla [6].

1.3.4 - Ejemplo: iie.fing.edu.uy.

A la última etiqueta en el nombre de los dominios se le denomina *host name.* Generalmente encontramos el host WWW –*World Wide Web*–, cuya popularización responde simplemente a una convención para representar páginas Web. Sin embargo, cualquier etiqueta es válida como host, en este ejemplo esta es iie.

Figura 1.2: Ejemplo de Dominio

El dueño de un domino puede delegar a su preferencia todo lo que se encuentra a la izquierda de su nombre de dominio. A cada unidad delegada, se le da el nombre de *zona*. En la Figura 1.2 podemos suponer que la autoridad de uy. delegó la autoridad del nombre de dominio edu.uy. Si el dueño de dicho dominio decidiera crear un subdominio, este se encontraría a la izquierda del mismo.

1.4 - Zonas y Archivos de Zonas

El espacio de nombres se divide en zonas disjuntas. Cada zona contiene los nombres de dominio y recursos autoritativos de la misma. El comienzo de la zona está determinado por el punto de delegación o corte de la zona padre, extendiéndose por el árbol de nombres hasta los próximos puntos de corte y delegaciones de las zonas hijas. El nodo más alto en una zona, corresponde al punto de corte y pertenece al nombre de dominio al que se le realizó la delegación en la zona padre. Para distinguir este dominio de los restantes de la zona, se lo llama también ápex de la zona. Una zona termina cuando se delega la administración de un nodo inferior a otra persona u organización.

La información que define una zona se almacena en un Archivo de Zona –*zone file*–. Estos archivos contienen toda la información autoritativa de la zona y los cortes o delegaciones a zonas hijas. Los servidores de nombres son los responsables de almacenar los archivos de zona. Se dice que un servidor es autoritativo para la zona uy si éste tiene almacenado localmente un archivo de zona con toda la información de la misma. En general, existe una sola copia maestra de un archivo de zona, la cual es trasferida a los demás servidores de nombres autoritativos para la zona. Dada la importancia de esta información, es recomendable tener al menos dos servidores que sirvan cada zona, de manera de tener mayor garantía de que los datos estarán siempre accesibles.

Los archivos de zona tienen como función la traducción de nombres de dominio en entidades operacionales, estas son host que almacenan páginas web, servidores de correo, servidores de nombre, etc. Los datos que describen el contenido del archivo se pueden dividir en las siguientes categorías:

- Datos autoritativos para todos los nodos dentro de la zona.
- Datos autoritativos que definen el nodo superior de la zona −ápex de la zona−.
- Datos que describen sub-zonas delegadas −cortes sobre el fin de la zona−.
- Datos que permiten el acceso a sub-zonas de servidores de nombres −a veces llamados datos "glue"−.

Los primeros dos tipos de datos se diferencian de los demás por ser autoritativos. Estos tendrán mayor importancia en DNSSec, ya que son los datos que se firmarán digitalmente.

Todos estos datos se expresan utilizando −Resource Records− registros de recursos RR. Las zonas pueden ser totalmente descritas utilizando los RR y transferidas entre los servidores de nombres utilizando estos RR. Los RR más importantes que se pueden encontrar en un archivo de zona son:

- Descripción de la zona. Registro de Recursos SOA −Start of Authority−. Este RR es obligatorio para todos los archivos de zona y es autoritativo.
- Un host de una zona que es definido con una dirección IP, lleva por lo menos un registro A o AAAA si es IPv6. Si el host esta dentro de la zona, el servidor será autoritativo para dichos recursos, de lo contrario no.
- Información global sobre las zonas. El RR MX que describe el servidor que acepta mails para este dominio, el RR NS describe el servidor DNS autoritativo para dicho dominio. Ambos son autoritativos para la zona.
- Un archivo de zona incluye información acerca de la delegación de subdominios, utilizando un RR NS. Esta información también puede incluir registros adicionales conocidos como registros *glue*. Estos registros no son autoritativos, por lo que como veremos más adelante, no llevarán una firma digital asociada.

Los RRs que describen el nodo superior de la zona son primordiales para la gestión de la zona −son parte de los datos autoritativos−. El primero que encontramos en el archivo es el RR SOA, el cual describe los parámetros de gestión de la zona, ver Anexo B. Los siguientes RR, también obligatorios, son los RR NS del ápex de la zona. Estos listan los servidores de nombre −NS por sus siglas en Inglés Name Server− que sirven autoritativamente a la zona. Es recomendable que existan por lo menos dos servidores de este tipo. También podemos encontrar en los RRs MX la descripción de los servidores de correo que pueden brindar este servicio para su dominio.

Los puntos de delegación que describen los límites o fin de la zona son los RR NS. Estos nombran a los servidores de nombre de las sub-zonas. Estos RR NS solo se encuentran en el ápex de la zona −que fue delegada− donde sí son autoritativos y en los cortes de zona −delegaciones− donde no son autoritativos.

Cuando un servidor recursivo recorre el árbol de nombres buscando y extrayendo la información que necesita de los servidores de nombres autoritativos,

hace uso de las delegaciones −los RR NS− para saber por qué partes del árbol debe seguir buscando. Un servidor de nombres conoce la ubicación −direcciones IP− de los servidores raíz, los cuales contienen la información necesaria para que el servidor recursivo siga su camino. Esta información no solo es el RR NS, sino que debe incluir también la dirección −RR A− de dicho servidor. A este registro se lo conoce como registro glue, y como ya mencionamos, no es autoritativo para la zona.

1.5 - Registros de Recursos

Un nombre de dominio identifica a un nodo del árbol DNS, que puede tener o no, asociado un conjunto de información definida en RRs. Cada RR está compuesto por una quíntupla de campos.

Tabla 1-1: Campos de un RR

Campo	Descripción
Propietario	Es el nombre de dominio donde se encuentra.
Tipo	Es un valor codificado de 16 bit que especifica el tipo de recurso. Los tipos se refieren a recursos abstractos. **A** Una dirección de host IPv4. **AAAA** Una dirección de host IPv6. **CNAME** Identifica a un nombre canónico de un alias. **MX** Identifica a un servidor de correo para el dominio. **NS** El servidor de nombres autoritativo para el dominio. **PTR** Puntero a otra parte del espacio de nombres de dominio. **SOA** Identifica el comienzo de la zona de autoridad.
Clase	Valor codificado de 16 bits que identifica a una familia de protocolos o una instancia de un protocolo. **IN** Internet **CH** El sistema Caos.
TTL	Tiempo de vida del RR. Este campo es un entero de 32 bits y representa segundos. Fundamentalmente lo utilizan los servidores recursivos cuando cachean RRs. El TTL describe el tiempo de duración que puede estar cacheado un RR antes de ser eliminado.
RDATA	Son los datos contenidos en el RR, dependen del tipo y de la clase. **A** Para la clase IN, dirección IP de 32 bits −IPv4−. Para la clase CH, un nombre de dominio seguido de una dirección Caos octal de 16 bit. **CNAME** Un nombre de dominio. **MX** Un valor preferentemente de 16 bit seguido de un nombre de host, que actúe como servidor de correo para el dominio propietario. **NS** Un nombre de host. **PTR** Un nombre de dominio. **SOA** Varios campos.

El nombre propietario es a menudo implícito, en vez de formar parte integral del RR. Las siguientes partes del RR se componen el encabezado −TTL, tipo, clase− y

por último, la parte variable donde se encuentra la información propia del RR es el campo RDATA. El contenido de este último campo, depende del tipo de RR, puede ser una combinación de strings binarios y nombres de dominio. Los nombres de dominio con frecuencia se utilizan como "punteros" a otros datos en el DNS.

En el Anexo B se puede encontrar ejemplos de RR, como se expresan y como se define el contenido de los mismos.

1.6 - Wildcard

Las definiciones, usos y funciones de las wildcard en el DNS estan especificadas en la RFC 4592 [7]. Aquí rescataremos las principales definiciones que nos serán de utilidad en el entendimiento de DNSSec.

Los RRs cuyos nombres de dominio comienzan con la etiqueta wildcard –"*"– tienen un tratamiento especial. Estos RRs se conocen como wildcards y representan instrucciones para sintetizar otros RRs. Este tipo de recurso especial es utilizado para permitir la resolución de nombres no existentes en el archivo de zona, pero que se pueden sintetizar utilizando un nombre de dominio wildcard.

Los contenidos de los RRs wildcard siguen las reglas usuales y formatos de los RR, que están disponibles en el Anexo B. El nombre de los RR wildcard tienen la forma "*".<cualquierdominio>, donde <cualquierdominio> es cualquier nombre de dominio. <cualquierdominio> no debería contener otras etiquetas "*" y debe pertenecer a la zona donde se encuentra el wildcard. Los wildcard se aplican potencialmente a descendientes de <cualquierdominio>, pero no para <cualquierdomino> en sí mismo.

La etiqueta "*" no tiene un efecto especial si aparece en un nombre de consulta, pero puede utilizarse para comprobar la existencia de un registro wildcard en una zona autoritativa; una consulta como esa es la única forma para conseguir una respuesta que contenga RRs con un nombre de propietario con "*" en él, el resultado de tal consulta no se cacheará. Los contenidos de los RRs wildcard no se modifican cuando se utilizan para sintetizar RRs.

1.7 - Tipos de servidores DNS

En esta sección se describirán los tipos de servidores DNS más comunes e importantes para el sistema. En particular se definirán los servidores autoritativos –maestro y esclavos– y recursivos. Estos son los tipos que tienen mayor importancia para nuestro proyecto. El resto de las posibles configuraciones de estos servidores se dejan como lectura complementaria en el Anexo D.

Los servidores DNS juegan una gran variedad de roles, un único servidor de nombres puede ser un maestro para algunas zonas, un esclavo para otras y proporcionar almacenamiento en cache o servicios de reenvío para otras.

Utilizaremos los comandos y archivos de configuración de BIND –Berkeley Internet Name Domain– para ilustrar los ejemplos y configuraciones de los servidores. BIND fue la herramienta que elegimos para crear los servidores en nuestro proyecto, ya que esta es la implementación más difundida de un servidor de nombres y cuenta con mayor soporte técnico en la red. Por más información de comandos y configuración ver Anexo F.

El rol del servidor de nombres es controlado por el archivo de configuración, en el caso de BIND se llama *named.conf*. La combinación de los parámetros globales en este archivo y las zonas que atiende determinan la funcionalidad completa del servidor de nombres. Se utilizarán fragmentos de este archivo –named.conf– a modo de ejemplos para desglosar las características de cada tipo de servidor.

1.7.1 - Servidor de Nombres Autoritativo

Los servidores autoritativos son los que poseen una copia local de uno o varios archivos de zonas, y responden de forma autoritativa por consultas hechas por dominios dentro de esas zonas.

La arquitectura más común en BIND de estos servidores es la conocida como maestro-esclavo. Los términos maestro y esclavo de zona se utilizan para diferenciar a los servidores de nombres que intervienen en la transferencia de zona. En la Figura 1.3 se visualiza el relacionamiento entre maestro y esclavo.

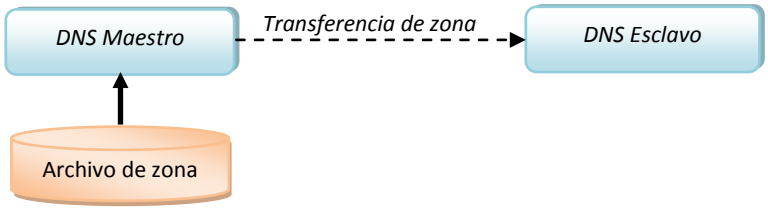


Figura 1.3: Transferencia de zona

Un servidor maestro se identifica por tener la copia maestra del archivo de zona. El servidor esclavo solicita una transferencia de zona cuando este reconoce que sus datos no están actualizados. La transferencia puede consistir en enviar toda la información del archivo de zona maestro o solo las partes modificadas. Entre los métodos para realizar una transferencia se encuentra el AXFR y el IXFR –Anexo E– Una vez realizada la transferencia de zona, los servidores esclavo y maestro son indistinguibles desde el punto de vista del cliente, ya que ambos responden autoritativamente por los dominios de la zona.

En el Ejemplo 1-1 tenemos una clausula de zona, en la que se define al servidor como maestro para la zona example.com. El archivo local donde se encuentra el archivo de zona está especificado usando la sentencia *file.*

Ejemplo 1-1: Configuración servidor maestro

```
//Define al servidor como maestro para example.com
zone "example.com" {
            type master;
            file "master.example.com";
};
```

El servidor esclavo es definido en BIND mediante la inclusión del *type slave* en la cláusula de la zona del archivo named.conf, como se muestra en el Ejemplo 1-2. El tipo *slave* indica que este servidor de nombres actuará como un esclavo para example.com.

Ejemplo 1-2: Configuración servidor esclavo

```
// Fragmento de named.conf
// Define un servidor esclavo para example.com
zone "example.com" {
            type slave;
            masters {192.168.23.17;};
};
```

La sentencia "*masters* {192.168.23.17}" define la dirección IP del servidor de nombre que contiene el archivo de zona maestro para esta zona. Una o varias direcciones IP pueden estar presentes. Puede haber más de un maestro DNS para cualquier zona. Desde el lado del servidor maestro se puede –y es aconsejable– restringir las direcciones IP a las cuales les está permitido realizar transferencia de zonas. No hacerlo puede significar potenciales ataques DoS –Negación de Servicio, que veremos más adelante–.

1.7.2 - Servidores de nombres recursivos

Un servidor de nombres recursivo –también conocido como resolver de cache, cache de DNS o resolver– obtiene su información mediante consultas a servidores de nombres autoritativos –vistos anteriormente–, con el fin de responder una consulta de un host o cliente, guardando además estos datos en un cache local. En una solicitud posterior por los mismos datos, el servidor de cache va a responder con los datos almacenados en cache, pero esta vez sin autoridad. Este proceso continuará hasta que el tiempo de vida –TTL– del RR llegue a cero, momento en que el RR será descartado de la memoria cache. La siguiente solicitud a este RR dará lugar a que el servidor recursivo nuevamente consulte a un servidor de nombres autoritativo para la zona. Los caches aumentan considerablemente el rendimiento de DNS reduciendo significativamente la carga sobre la red y los servidores autoritativos. Consideremos el Ejemplo 1-3, donde se definió el servidor de correo de example.com:

Ejemplo 1-3: MX con cache

```
example.com. 3w   IN    MX    10    mail.example.com.
```

En el Ejemplo 1-3 se define el RR MX para example.com, cuyo TTL es de tres semanas –3 weeks por su sigla en Inglés–. Una vez obtenido este registro el cache lo almacenará durante 3 semanas, respondiendo directamente de cache cuando una

consulta por el mismo le llegue. De esta forma se alivia el tráfico hacia el servidor autoritativo evitando consultas excesivas por recursos que no varían significativamente con el tiempo, como este caso. Sin embargo, de ser necesario realizar un cambio a un registro de recurso con valores grandes de TTL, se tendrá que tener en cuenta que los cambios no se propagarán completamente en toda la red hasta que todos los cache hayan expirado los registros viejos.

En BIND, un servidor recursivo tendrá en su archivo named.conf el siguiente fragmento:

Ejemplo 1-4: Configuración servidor recursivo

```
// Fragmento de named.conf
// Configuración para permitir recursión, aunque esta es por defecto.
options {
        recursion yes;
        allow-recursion {10.2/16;192.168.2/24;}; // Redes que pueden utilizar
// la recursión.
};

// Cláusula de zona

//El punto indica la zona raíz
zone "." {
     type hint;
     file "root.servers";
};
```

La cláusula de opciones indica que las declaraciones siguientes se aplican a todas las zonas, a no ser que explícitamente sea remplazada con otra declaración. La sentencia "*recursion yes"* indica el comportamiento de cache, que es el valor por defecto de BIND y podría ser omitido. La declaración "*allow-recursion* {10.2/16; 192.168.2/24;}" define las direcciones IP –redes– que están autorizadas a emitir consultas recursivas. Si esta declaración no está presente, esto podría ser un servidor de nombres *caching open* y como tal, podría ser utilizado por terceros maliciosos en ataques DoS.

Cuando llega una consulta al servidor recursivo, este debe saber donde realizar la primera consulta, y lo más razonable y eficiente −si no se tiene información del nombre de dominio consultado en cache o de algún acsendente− es que pregunte primero en la zona raíz. Para esto necesita tener configuradas las direcciones de red donde se encuentran los servidores de nombre autoritativo para la zona raíz. Mediante la clausula de zona del Ejemplo 1-4, se indica que el archivo conteniendo dichas direcciones −root.servers−. Este archivo se suministra normalmente con distribuciones de BIND y demás software. También puede encontrarse en Internet en [8].

1.8 - Operaciones DNS

Algunas operaciones del DNS ya las hemos mencionado, las transferencias de zonas y las consultas son algunas de ellas. Aquí nos centraremos específicamente en

los tipos de consultas que el protocolo permite y cómo funcionan, para ver luego sus diferencias al introducir DNSSec.

Por defecto las consultas utilizan UDP y el puerto 53. En los inicios del DNS las consultas tenían un tamaño máximo de segmento UDP de 512 bytes. Pero a medida que fue creciendo la infraestructura del DNS fue también en aumento el volumen de datos en las transacciones DNS. En los casos en que una respuesta superara este límite, se utilizaba TCP para realizar la transacción, pero debido a su overhead, esta opción no es la favorita. Para evitar usar TCP cuando el bloque de transacción supera los 512 bytes, se planteo una funcionalidad conocida como *Extended DNS*, cuya función es esencialmente negociar un segmento UDP de tamaño extendido. La mayoría de las transacciones DNS no seguras se pueden realizar con mensajes de tamaño menor a 512 bytes. Sin embargo, la implementación de DNSSec tiene como consecuencia un bloque en el rango comprendido entre 1.7K y 2.4K, o incluso mayor, lo que hace necesaria la negociación de bloques más grandes.

1.8.1 - Consultas

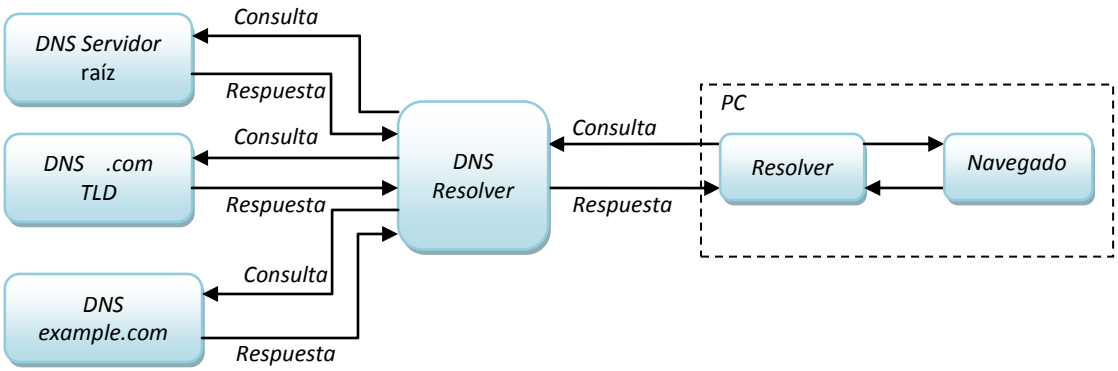


Figura 1.4: Consulta recursiva

Las consultas DNS las podemos clasificar según las siguientes categorías.

- *Consultas recursivas:* Una consulta recursiva es aquella en la que el servidor hará todo el trabajo necesario para devolver la respuesta completa a la consulta. La respuesta a una consulta recursiva puede hacer que el servidor de nombres envíe múltiples consultas a una serie de servidores de nombres autoritativos en la jerarquía DNS, a fin de resolver completamente el nombre solicitado. Los servidores de nombres no están obligados a soportar consultas recursivas.
- *Consultas Iterativas –o no recursivas–:* En una consulta iterativa, si el servidor de nombres tiene la respuesta o si se encuentra disponible en su cache la devolverá. Si el servidor de nombres no tiene la respuesta devolverá toda la

información que posea, en general una referencia al siguiente nivel de delegación, pero no va a hacer solicitudes adicionales a otros sistemas de servidores de nombre. Todos los servidores de nombres deben ser compatibles con las consultas iterativas.

- *Consultas inversas:* El usuario quiere saber el nombre de dominio dado un RR[1]. Los servidores de nombres no estaban obligados a admitir las consultas inversas y la característica rara vez se aplicó. Finalmente RFC 3425 lo declaró obsoleto.

1.8.1.1 - Consulta Recursiva

Una consulta recursiva es cuando el servidor recursivo responde totalmente la consulta −o da un error−. Los servidores de nombres autoritativos no necesariamente deben soportar las consultas recursivas. Hay tres posibles respuestas a una consulta recursiva:

1. La respuesta a la consulta acompañada de todos los registros CNAME −alias− que pueden ser útiles. La respuesta siempre indicará si los datos vienen desde un servidor autoritativo o desde el cache.
2. Un error que indica que el dominio o el host no existe −NXDOMAIN−. Esta respuesta también puede contener los registros CNAME que apuntan al host que no existe.
3. Un error de indicación temporal.

En una consulta recursiva, un servidor de nombres seguirá la ruta de delegación establecida por los servidores de nombres autoritativos, hasta obtener la respuesta a la consulta. Pondremos como ejemplo el caso de la Figura 1.4 donde nos encontramos con que se intenta resolver la dirección IP de example.com:

1. En el primer paso el stub-resolver consulta al servidor DNS local el cual intentara resolver la consulta dado que éste está configurado para permitir consultas recursivas. En principio el servidor recursivo buscará la dirección en su tabla local o cache antes de comenzar a realizar las iteraciones.
2. Suponiendo que no tenemos la respuesta en el cache, el servidor comenzará enviando la consulta a un root-server, quienes no admiten consultas recursivas, por lo que su respuesta será una lista de servidores de nombres que sean autoritativos para el siguiente nivel en la jerarquía de nombres de dominio –en este caso el gTLD com−.
3. Se seleccionará uno de los servidores de la lista y se le enviará la consulta, estos servidores solo permiten consultas iterativas por lo que nos devolverá otra referencia al dominio inferior siguiente.

[1] Resource Records, registros de datos DNS.

4. El servidor recursivo selecciona uno de los servidores autoritativos para example.com de la referencia anterior y envía una consulta para obtener la IP −el RR A − de www.example.com.
5. Al recibir la respuesta a la consulta el DNS local enviará la respuesta al usuario –en este caso el DNS local realiza toda la iteración por el usuario−.

El *stub-resolver* en la PC del usuario siempre envía una consulta recursiva −una consulta que le pide al receptor que actúe de forma recursiva y retorne la respuesta completa−. Este servidor acepta la consulta recursiva y comienza el proceso de envío de múltiples consultas a lo largo de la jerarquía DNS para resolver la consulta. El servidor recursivo típicamente también envía una consulta solicitando servicios recursivos a todos los servidores autoritativos. Los root-servers autoritativos y servidores gTLD nunca soportan recursión, así que en esos casos estos solo devuelven una referencia al siguiente nivel de jerarquía. Se considera una buena práctica que un servidor de nombre de dominio autoritativo sea configurado para no soportar recursión.

Queda por ver como se decide qué servidor utilizan los DNS cuando se tiene a disposición múltiples servidores de nombres como los casos de *root-servers* y *gTLD*. En estos casos la mayoría de los servidores de nombres utilizan algún algoritmo para distribuir la carga y entonces asegurar el resultado más rápido posible.

1.8.1.2 - Consultas iterativas −no recursivas−

En una consulta iterativa −o no recursiva− el servidor de nombres puede proveer una respuesta parcial a la consulta −o dar un error−. Todos los servidores de nombres deben soportar consultas no recursivas.

Hay cuatro posibles respuestas a consultas no recursivas:

1. La respuesta a la consulta acompañada por registro CNAME –alias−. La respuesta debe indicar si los datos son autoritativos o de cache −no autoritativo−.
2. Un error que indique que el dominio o el host no existe –NXDOMAIN−. Esta respuesta también puede contener los registros CNAME que apuntan al host que no existe.
3. Un error de indicación temporal.
4. Una referencia: uno o más servidores de nombres que están más cerca del siguiente nivel inferior en la jerarquía de nombres para el nombre de dominio solicitado. Pueden o no ser los servidores de nombres autoritativos para el dominio final de la consulta. Una referencia es el método de respuesta normal utilizado por servidores raíz y los servidores de dominio de nivel superior.

Ahora veamos el camino de una consulta a un servidor de nombres que no soporta consultas recursivas. La resolución de una consulta por www.example.com. se resuelve siguiendo el esquema de la Figura 1.5:

1. Primero nos encontramos con que el host enviará la consulta al servidor local, el cual buscará www.example.com en la tabla local o cache, de tener esta información allí contestará la consulta con la información que posea e indicando que no es una respuesta autoritativa.
2. Dado que supondremos no tiene la respuesta en el cache ni es autoritativo para este dominio y debido a que esta consulta es iterativa –no solicita recursividad–, el servidor responde con una referencia que contiene la lista de servidores raíz.
3. El servidor recursivo selecciona uno de los servidores raíz y envía la consulta directamente a este. Luego el servidor raíz responde con una lista de servidores de nombres autoritativos para el *gTLD* com –otra referencia–.
4. El servidor selecciona uno de los servidores autoritativos *gTLD* devuelto en la referencia y envía una consulta por la IP de www.example.com directamente a ese servidor de nombres. El servidor de nombres *gTLD* responderá al servidor recursivo con los servidores de nombres autoritativos para el *SLD* example.com.
5. El servidor selecciona uno de los servidores de nombres autoritativos SLD devuelto en la referencia y directamente le envía una consulta. El servidor devolverá la respuesta autoritativa para ese dominio al host.

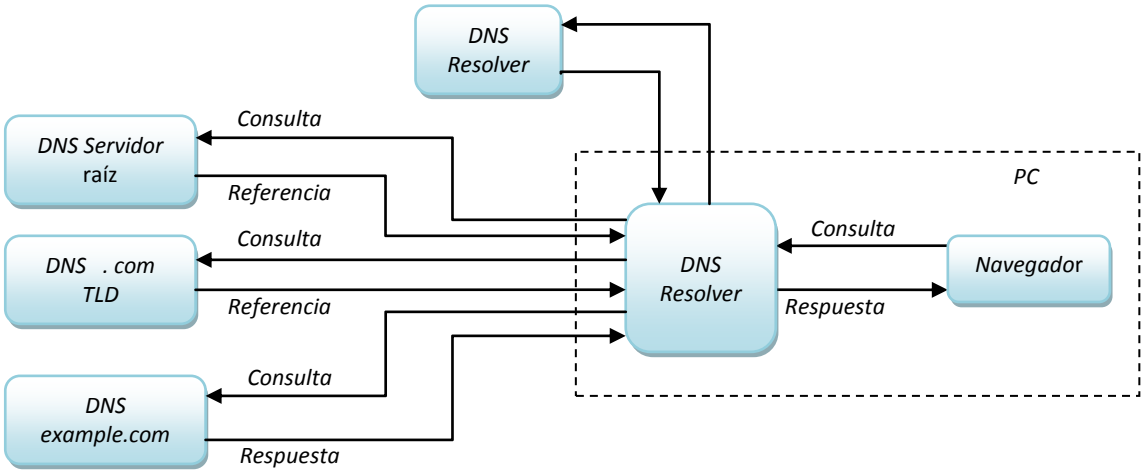


Figura 1.5: Consulta Iterativa

1.8.2 - EDNS

El DNS como fue implementado en su comienzo, en la RFC 882 [1], no previó la necesidad usar paquetes UDP de tamaño mayor a 512 octetos. El formato del mensaje junto con los formatos estándares para las opciones de codificación, errores y

compresión de nombres, fueron establecidos por la RFC 1035 [4], donde también se especifica un tamaño máximo de paquete fijo el cual resultó ser muy pequeño para los protocolos más recientes.

Dado los requerimientos actuales, se planteo la necesidad de un mecanismo de extensión que permitiera a los solicitantes anunciar el tamaño de los paquetes UDP que pueden soportar, de forma de permitir la transferencia de paquetes UDP de tamaño mayor a 512 octetos.

1.8.2.1 - OPT pseudo-RR

Cabe preguntarse cómo funciona el mecanismo para que un servidor DNS reconozca si el solicitante soporta EDNS. Aquí es donde ingresa en la discusión el OPT pseudo-RR. El servidor DNS debe recibir una consulta que contenga un registro de recursos OPT. Un registro OPT no contiene datos DNS reales y su contenido está relacionado únicamente con mensajes de la Capa de transporte UDP. El registro OPT indica el tamaño máximo permitido de la carga de UDP del emisor en su campo CLASS y muestra el número de octetos de la carga UDP máxima.

Cuando el servidor DNS reciba una consulta que contenga un registro OPT que anuncie el tamaño máximo del paquete UDP, truncará cualquier tamaño de respuesta UDP mayor que el límite especificado en el registro OPT. La presencia de un pseudo RR OPT en una consulta deberá ser tomada por quien la recibe como indicación de que quien la envía implementa completamente EDNS y soportará cualquier respuesta que cumpla con las especificaciones EDNS.

Si el servidor DNS recibe una consulta que no contiene un registro de recurso OPT, supone que el servidor del solicitante no admite EDNS y responderá al solicitante asumiendo que el emisor no acepta paquetes UDP mayores de 512 octetos. En este caso, el servidor DNS truncará su tamaño de respuesta UDP a un máximo de 512 octetos.

Los servidores que no soportan esta extensión de protocolo deberán responder con RCODE NOTIMPL, FORMERR, o SERVFAIL. Si el servidor soporta EDNS esta información puede ser guardada en cache por el solicitante de información aunque se debe consultar periódicamente para chequear que los valores sigan siendo válidos.

Según la RFC 4035, un servidor security-aware deberá soportar la extensión de mensaje EDNS0 soportando un tamaño de mensaje mínimo de 1220 octetos y debería ser capaz de soportar mensajes de hasta 4000 octetos. Así también en el caso de que se utilice IPv6, como los paquetes solo podrán ser fragmentados en el host que los ensambla, el servidor deberá asegurarse que los datagramas que trasmite con IPv6 estén fragmentados.

Un OPT pseudo-RR se puede añadir a la sección de datos adicional de una consulta o una respuesta. Cabe aclarar que se le denomina pseudo-RR porque no

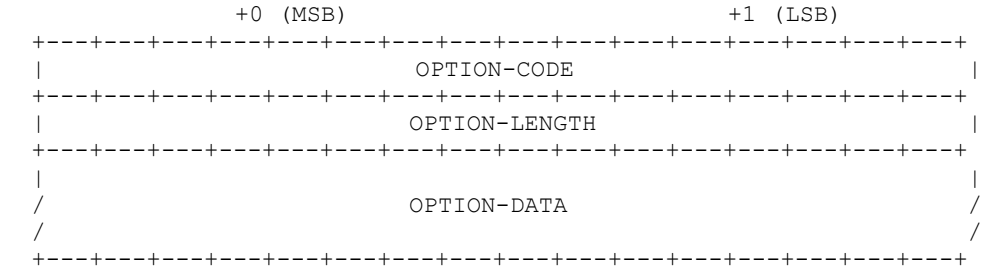
contiene datos reales de ningún RR, sino que trae información de capa de transporte. El RR OPT nunca se deberá mantener en caché, ser transmitido, almacenado o cargado desde archivos maestros. La cantidad de OPT pseudo-RR en cada mensaje podrá ser cero o uno, no mayor.

Un RR OPT tiene una parte fija y un set variable de opciones expresado como pares {Atributo, Valor}. La parte fija contiene algunos datos DNS así como también una pequeña colección de otros protocolos.

La parte fija de un RR OPT está estructurada de la siguiente manera:

```
Field Name Field Type Description
------------------------------------------------------
NAME domain name empty (root domain)
TYPE u_int16_t OPT
CLASS u_int16_t sender's UDP payload size
TTL u_int32_t extended RCODE and flags
RDLEN u_int16_t describes RDATA
RDATA octet stream {attribute, value} pairs
```

Mientras que la parte variable esta codificada en el RDATA y está estructurada de la siguiente manera:




```
OPTION-CODE (Assigned by IANA.)
OPTION-LENGTH Size (in octets) of OPTION-DATA.
OPTION-DATA Varies per OPTION-CODE.
```

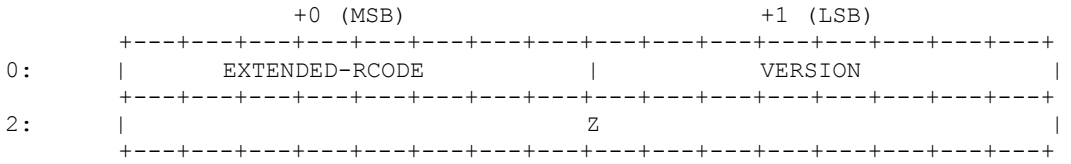
El tamaño de carga útil del usuario –que es guardado por OPT en el RR CLASS– es el número de octetos de la máxima carga útil del paquete UDP que puede ser re-ensamblado y entregado en el stack de la red del usuario. El MTU del camino con o sin fragmentación podría ser menor que esto.

Una carga útil de 512 bytes UDP requerirán un buffer de re-ensamblado de 576 bytes a nivel de IP. Se debe tener esto en cuenta a la hora de la elección a nivel de Ethernet ya que la consecuencia de la elección de un valor muy grande puede ser un mensaje ICMP de un Gateway intermedio –en general se toma 1280 como un valor razonable–.

Es recomendable tanto para quien envía como para el destinatario, tomar en cuenta el MTU del camino a la hora de considerar los tamaños del mensaje, ya que de no hacerlo se perderá eficiencia debido a la fragmentación de paquetes.

Debido al overhead se desaconseja anunciar el límite arquitectural como el límite máximo del payload UDP. Solo porque el stack pueda re-ensamblar 64Kb no se debe asumir que se deseara utilizar más de 4Kb de memoria por transacción. El anunciar un límite arquitectural también es contraproducente al plantearse la seguridad del servidor, nuevos problemas de ataques de Denegación de Servicio –ver sección 2.3.2.1 - ya que los servidores podrían enviar respuestas muy grandes llegando a producirse una tormenta de mensajes ICMP.

El RCODE extendido y las flags que OPT almacena en los campos RR TTL están estructurados de la siguiente manera:

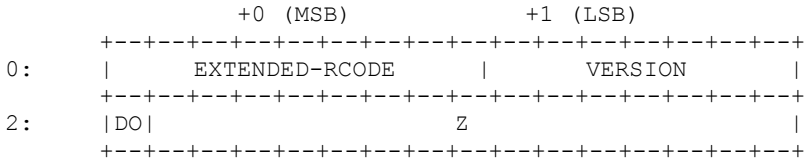


EXTENDED-RCODE
Forma los 8 bits más significativos del RCODE extendido. Si se utiliza un EXTENDED-RCODE "0" indicara la utilización de un RCODE sin extensión.

VERSION
Se utiliza para indicar el nivel de implementación, la versión "0" indica plena conformidad con la RFC 2671.
Si quien responde no soporta el nivel de versión de quien consulta responde con un RCODE=BADVERS.
Todas las respuestas se limitan en formato al nivel de la versión de quien realiza la solicitud. Como efecto secundario quien realiza la solicitud podrá conocer el nivel de implementación de quien responda.

Z
En la RFC 2671 quienes la envían lo setean a cero y es ignorado por quien lo recibe, en dicha RFC se plantea la posibilidad de ser modificado en subsiguientes RFC.

El mecanismo elegido para la notificación explicita de la habilidad de aceptar o entender RR de DNSSec, es usar el bit más significativo del campo Z en el encabezado OPT de EDNS0 de la consulta. Se hace referencia [8] a este bit como "DNSSec OK" –bit DO–.



Si el bit DO esta seteado en una consulta, le indica al servidor que quien la envía es capaz de soportar DNSSec, en caso de no tenerlo seteado significará que éste no podrá soportar estos registros y no deberán ser devueltos en la respuesta a menos que se soliciten explícitamente. Los DNSSec aware resolvers no deberán insertar los RRSIG, DNSKEY, NSEC RRs para autenticar la respuesta a menos que el bit DO este seteado en la consulta.

Un DNSSec aware server debe setear el DO bit en las consultas recursivas sin importar el estado inicial del DO bit en la consulta inicial. En el caso en que la consulta inicial no tenga el bit seteado el servidor recurivo deberá remover los registros DNSSec antes de enviarle la respuesta al cliente.

2. Introducción a la seguridad en DNS

La operación de los servidores de nombre de dominio –DNS– abre la puerta a amenazas potenciales a la seguridad. Como en el caso de cualquier otro servicio de acceso público, por ejemplo un sitio web o sitio FTP.

El punto crítico en la definición de políticas y procedimientos de seguridad es entender lo que hay que garantizar –o más bien de qué nivel de amenazas necesitamos estar protegidos y que nivel de amenaza es aceptable–. Las respuestas a estos dos puntos serán muy diferentes si el DNS está funcionando como root-server o corriendo DNS para un par de sitios web de bajo volumen.

2.1 - Amenazas a la seguridad

Para poder evaluar correctamente tanto las amenazas como las contramedidas a tomar, es necesario observar primero el flujo normal de datos en un sistema DNS:

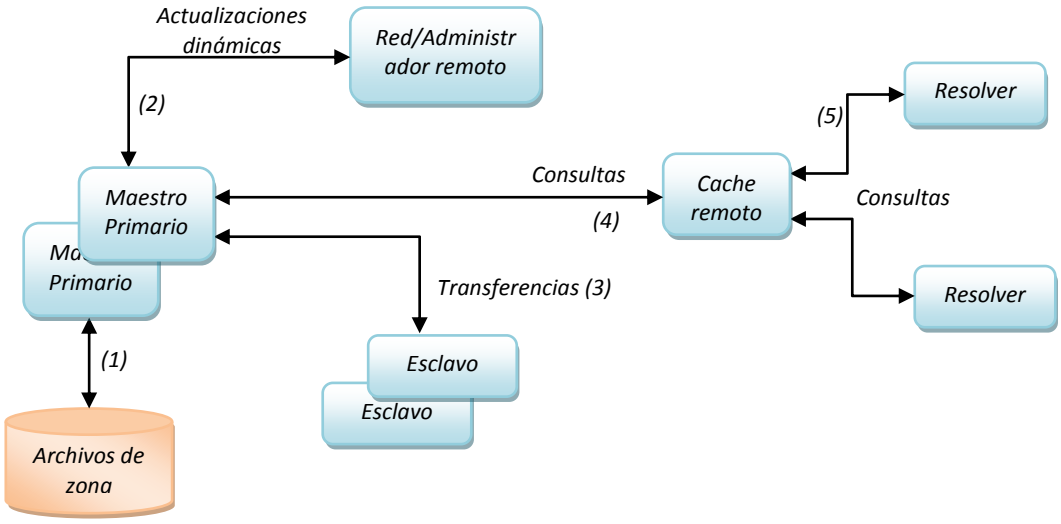


Figura 2.1: Flujo DNS

Tabla 2-1: Amenazas y soluciones

Numero	Área	Amenaza	Clasificación	Solución
1	Archivos de zona.	Corrupción de archivos –tanto maliciosa como accidental–.	Local.	Administración del sistema.
2	Actualizaciones automáticas.	Actualizaciones no autorizadas.	Servidor-Servidor.	Arquitectura de la red, TSIG, SIG(0) o deshabilitar.
3	Transferencias de zona.	Spoofing de direcciones IP	Servidor-Servidor.	Arquitectura de la red, TSIG, TKEY o deshabilitar.
4	Consultas remotas.	Envenenamiento de cache, intercepción de datos o corromper un maestro o esclavo.	Servidor-Cliente.	DNSSec.
5	Resolución de consultas.	Envenenamiento de cache, intercepción de datos, corromper un maestro o esclavo, IP spoofing.	Remoto Cliente-Cliente.	DNSSec, SSL/TLS.

Cada interacción de esta figura que esta numerada, representa una amenaza potencial. En la Tabla 2-1 se pueden observar las posibles amenazas y posibles soluciones.

La primera fase de una revisión de seguridad es el control de que amenazas son aplicables al sistema en cuestión y cuan serias pueden volverse, estas se clasifican en cada caso en particular. Por ejemplo, si las actualizaciones dinámicas no son soportadas, no hay ninguna amenaza de actualización dinámica. Por lo que puede ser más fácil desactivar un proceso que implementar las medidas para la seguridad del mismo, o de lo contrario, puede ser más sencillo elaborar sistemas de seguridad que cambiar a otros procesos.

A modo de ejemplo, es posible desactivar las trasferencias de zona a servidores esclavos, lo que causaría que las actualizaciones se tengan que hacer de forma manual. Esto evita que usuarios ajenos a la administración de la zona puedan descargar la misma, disminuyendo los riesgos de seguridad en DNS. En ocasiones esto se denomina como seguridad por oscuridad.

Por último, cuanto más lejos se vaya del archivo de zona maestro, es más complicada la solución y la ejecución. A menos que haya una muy buena razón para no

hacerlo se recomienda siempre partir desde la zona maestra y trabajar hacia el exterior.

2.2 - Clasificación

Una clasificación para los riesgos a la seguridad se puede encontrar en [10], estas clasificaciones son utilizadas como medio para permitir la selección de los recursos y estrategias adecuadas para evitar estos riesgos. La numeración de la lista siguiente se refiere a la Figura 2.1**.**

1. Amenazas locales −1−: Las amenazas locales suelen ser más simples de prevenir, por lo general requieren de buenas políticas de administración del sistema. Los archivos de zona y los archivos de configuración de DNS −named.conf en el caso de BIND− contienen una gran cantidad de datos que deben estar respaldados y se debe tener derechos de lectura y escritura limitados.
2. Servidor-Servidor −2−: Si una organización corre servidores de nombre esclavos esta necesitará realizar transferencias de zona. Es posible ejecutar múltiples servidores de nombres maestro en lugar de servidores maestro-esclavo con lo que se podrían evitar los problemas asociados a la configuración maestro-esclavo. Si se requiere realizar la transferencia de zona TSIG y TKEY también ofrecen métodos de seguridad criptográfica para la autenticación de las fuentes y destinos de la solicitud. La transferencia física puede ser asegurada utilizando Secure Sockets Layer −SSL− o Transport Layer Security −TLS−.
3. Servidor-Servidor −3−: Una solución es negar las actualizaciones dinámicas de la zona. El diseño de la red, toda la arquitectura de los sistemas involucrados en un perímetro de confianza, puede reducir aún más la exposición. TSIG y SIG (0) se puede utilizar para asegurar las transacciones.
4. Servidor-Cliente −4−: Los caches en los servidores recursivos pueden llegar a ser “envenenados”, con el objetivo de por ejemplo modificar su contenido para apuntar a sitios web de los competidores "IP spoofing”, intercepción de datos y otros hacks. Aunque los sitios web modestos, probablemente tienen poco que temer de esta forma de ataque, si el sitio es de alto perfil, alto volumen, abierto a la amenaza de la competencia, o una fuente de ingresos altos, los costos y la complejidad de implementar una solución de seguridad a gran escala valen la pena.
5. Cliente-Cliente −5−: Las consultas clientes-cache deben también proporcionar mecanismos de seguridad que permitan autenticar las respuestas almacenadas por los mismos.

Nuestro trabajo se centrará en estos dos últimos puntos, los cuales pueden ser asegurados utilizando la extensión de seguridad DNS. Veremos a continuación las

vulnerabilidades en DNS, y nos concentraremos en aquellos aspectos que conciernen a DNSSec.

2.3 - Vulnerabilidades del protocolo DNS

Veamos aquí una clasificación general [11] del tipo de vulnerabilidades y amenazas conocidas en DNS. Cada una de estas categorías contiene ataques que son solucionables utilizando DNSSec, mientras que otros son amplificados al utilizar esta extensión, como los ataques DoS.

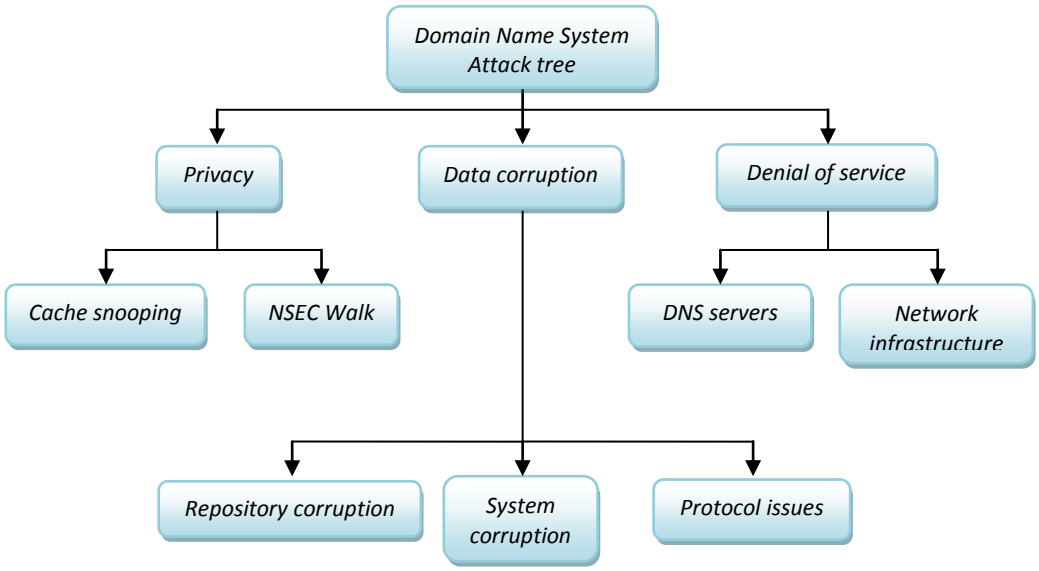


Figura 2.2: Árbol de ataques Dns

La primera de las categorías principales de amenazas a la seguridad en DNS es la corrupción de datos. Esta se define como todo tipo de incidentes relacionados con la modificación no autorizada de datos DNS. Estos incidentes pueden suceder en cualquier momento, en cualquier parte de la cadena de propagación de DNS. Dentro de esta categoría, veremos más en detalle los que se denominan *Protocol Issues,* que tienen que ver con los abusos y explotaciones de las deficiencias en el protocolo DNS y sus implementaciones.

2.3.1 - Protocol Issues

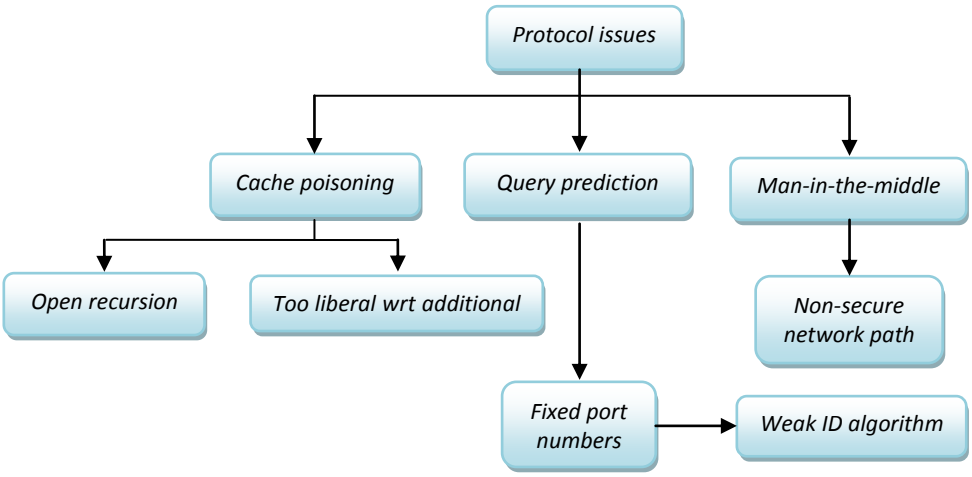


Figura 2.3: Problemas de protocolo

Entre estos ataques encontramos los quizás más conocidos, como *Cache poisoning* y *Man in the Middle.* Otros quizás no tan conocidos pero aún vigentes son los relacionados con las predicciones de consultas.

2.3.1.1 - Envenenamiento de cache

El envenenamiento de cache es cuando un atacante logar que se guarde en cache −ya sea en el equipo final o en el servidor recursivo de la red− un registro falso. La corrupción de un servidor de nombres tiene como objetivo generalmente la sustitución de una dirección de Internet verdadera por otra dirección falsa. Cuando un usuario ingresa a una web por medio de su dirección, la solicitud se redirige por la entrada en la tabla a una dirección diferente. En ese momento un gusano, spyware, hijacking, u otro tipo de malware puede ser descargado al ordenador del usuario.

El envenenamiento de la cache puede realizarse mediante una variedad de maneras, aumentando la velocidad a la que programas maliciosos se propagan. Una de las tácticas más usuales es la colocación de URLs comprometidas en mensajes de correo electrónico, spam, con líneas de asunto que tientan a los usuarios a abrir el mensaje. Las imágenes y los anuncios dentro de los mensajes de correo electrónico también pueden ser vehículos por los cuales los usuarios son dirigidos a los servidores que han sido comprometidos por el envenenamiento del cache. Una vez que la computadora de un usuario final ha sido infectada con el código nefasto, todas las solicitudes futuras por el equipo del usuario de la dirección URL atacada van a ser redirigidas a la dirección IP maliciosa. El envenenamiento de caché es especialmente peligroso cuando los objetivos son sitios de confianza bien conocidos, como aquellos a los que se dirigen los navegadores a la hora de realizar las actualizaciones automáticas de las definiciones de virus.

Existen varios métodos de envenenamiento de cache, algunos ya han sido mitigados por controles más estrictos en los mecanismos de almacenamiento en cache de los programas. Uno de los más sencillos de realizar consiste en realizar consultas al servidor objetivo −el que queremos envenenar− por dominios por los cuales somos autoritativos, y por lo que el servidor objetivo nos tendrá que preguntar.

Ejemplo 2-1: Pregunta maliciosa

```
Query: subdominio.midominio.com.   IN    A
```

Esto precisa que tengamos un registro –midominio− registrado en la zona "com" y apuntando a un servidor nuestro. Lo cual no es ninguna dificultad ya que por poca plata cualquier persona puede comprar un nombre de dominio.

El envenenamiento de cache consiste en que cuando el servidor objetivo nos pregunte a nosotros −servidor autoritativo de midominio.com. −, le agreguemos a la respuesta información adicional −el envenenamiento− para que el servidor recursivo guarde en cache.

Ejemplo 2-2: Respuesta maliciosa

```
Respuesta:

Sección autoritativa:
midominio.com. IN   NS  dominioatacado.org

Sección adicional:
dominioatacado.org IN  A   a.b.c.d
```

De esta manera logramos redirigir las consultas por dominios dentro de la zona dominioatacado.org hacia nuestro servidor a.b.c.d. Los servidores de nombres han mitigado en gran parte este tipo de ataque, haciendo más estricta las condiciones necesarias para que un registro sea almacenado en cache. En este caso, dado que la sección adicional de la respuesta no pertenece a la zona de autoridad, este registro no sería almacenado. Esto se conoce como la *bailiwick rule*.

Otra variante de envenenamiento de cache, que hace uso de la predicción de preguntas, es falsificando respuestas DNS. Este ataque consiste en hacer demorar la respuesta real para poder enviar antes nuestra respuesta falsificada. Todas las consultas DNS tienen asociado un número identificador de 16 bits llamado *nonce*, que se utiliza para identificar las respuestas de cada consulta −la respuesta utiliza este mismo número−. Si el atacante puede predecir este número y enviar la respuesta antes de que llegue la verdadera habrá efectuado un envenenamiento de cache exitoso. Otra dificultad, además de predecir este número, es el puerto origen, el cual tiene que coincidir con el puerto destino de la respuesta. Las primeras versiones de BIND, elegían aleatoriamente un número nonce y luego lo incrementaban con cada consulta, lo que hacía muy sencillo predecir estos números. Actualmente, los servidores de nombre toman números aleatorios, tanto para el nonce como para el puerto, de manera de dificultar la predicción.

Con DNSSec esta vulnerabilidad deja de ser un problema, ya que quien responde a una consulta de forma autoritativa, estará obligado a incluir también las firmas digitales, lo que le permite al resolver rastrear la autenticidad de dicha respuesta hasta una fuente confiable: el trust anchor.

2.3.1.2 - Man-in-the-middle

Hombre en el medio –Man in the middle– es una descripción general de los ataques que se ejecutan cuando un atacante se encuentra entre dos hosts −por ejemplo, un servidor y un cliente−. El atacante tendrá por lo tanto conocimiento acerca de la conexión y puede usarlo para escuchar en la conexión o incluso inyectar datos en ella.

Esta clase de ataques puede ser causado −o ser causa− por envenenamiento de caché. En estos ataques generalmente el atacante se inserta entre la PC de la víctima y el servidor DNS. El atacante intercepta las respuestas a las consultas de las víctimas y envía información falsa mediante la asignación de direcciones incorrectas. Para que el ataque tenga éxito, el atacante tiene que enviar las respuestas antes de que el servidor DNS lo haga. Esto será posible dado que el servidor DNS podría estar realizando consultas recursivas o el atacante podría realizar ataques de denegación de servicio para relentizar el servidor DNS primario de la víctima. Una vez que la víctima tiene información incorrecta ésta es redirigida al sitio web malicioso.

2.3.2 - Denegación de servicio –Denial of Service–

Nuestra segunda categoría de ataques DNS es el ataque de negación de servicio. Estos ataques refieren a un tipo de ataque que hace que el servicio no esté operativo para los usuarios legítimos. Estos ataques están dirigidos a un servicio específico –como DNS– o más ampliamente dirigido a toda una parte de la red –Internet–.

Casi todos los ataques DoS, los más difíciles de mitigar, se basan en consumir tantos recursos como se pueda del objetivo, en este caso un servidor DNS. Esto se conoce como *Resource Starvation.*

Cuando un servidor de nombres recibe de forma accidental o intencional demasiadas peticiones, y la red no es un factor limitante, entonces es posible que el servidor permanezca totalmente ocupado con la recepción, y básicamente, no tenga tiempo para responder.

2.3.2.1 - Ataques DoS y DDoS

Un ataque de denegación de servicio se ejecuta desde un host atacante hacia el host de la víctima. El host atacante tratará de consumir la mayor cantidad de recursos

tanto de la víctima como de la infraestructura que lleva a la red de las víctimas, por lo que el servicio para los usuarios normales se degrada.

Como los servidores DNS están en general bien aprovisionados hoy en día −en términos de recursos−, este tipo de ataques no deberían afectar a Internet ya que el propio atacante sería de hecho un cuello de botella que limitaría la extensión del ataque.

Una de las principales dificultades que enfrenta el despliegue de DNSSec es este tipo de ataques, ya que incrementa la vulnerabilidad del sistema frente a estos ataques. Las respuestas DNSSec consumen más ancho de banda, ya que los registros RRSIG que contienen las firmas digitales son mucho más grandes que los propios registros, lo que incrementa el tráfico cursado. Además, no solo el ancho de banda se ve afectado, los servidores autoritativos y recursivos deben realizar también operaciones criptográficas en tiempo de operación al devolver consultas del tipo NXDOMAIN. Este nuevo comportamiento facilita enormemente el esfuerzo necesario para la realización de un ataque DoS, dejando en evidencia que una de las vulnerabilidades más importantes de DNSSec es la amplificación de estos ataques, como veremos más adelante en nuestras pruebas.

2.3.3 - Privacidad -Privacy-

Algunos problemas con el DNS no son tanto un problema de seguridad en el sentido del intercambio de datos. En su lugar, están más relacionados con la privacidad ya que permiten a un atacante conseguir datos DNS no accesibles.

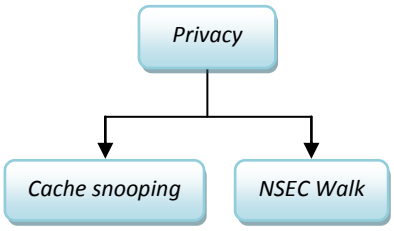


Figura 2.4: Privacidad

2.3.3.1 - Cache snooping

DNS cache snooping es el proceso de determinar si un registro de recursos –RR- esta −o no− presente en un cache DNS dado. Esto brinda información acerca de que consultas maneja un servidor.

En otras palabras DNS cache snooping es una técnica que permite conocer los nombres de dominio que han sido resueltos por un servidor DNS en particular. Esta información puede llegar a ser interesante ya que al atacante le permite crear un perfil de los patrones de uso de los usuarios de dicho servidor DNS.

2.3.3.2 - NSEC Walk

DNSSec incluye el RR NSEC para proporcionar la negación de existencia autenticada. Aunque el RR NSEC reúne los requisitos para la negación de la existencia autenticada introduce como efecto secundario que el contenido de una zona se pueda enumerar. Este problema le presentó dificultades al depliegue de DNSSec ya que no todos los administradores de zonas están dispuestos a permitir que sus datos sean enumerados. La enumeración de zona consiste en que un atacante pueda conocer todos los nombres de las maquienas de un dominio, lo cual puede ser contrario a los requerimientos de seguridad de los administradores.

3. Introducción DNSSec

3.1 - Historia

Antes de abordar temas específicos de DNSSec veamos un poco acerca de los orígenes de esta extensión, como se desarrollo a lo largo de los últimos años y en qué punto del despliegue DNSSec estamos actualmente.

Si bien no se puede hablar de una fecha exacta en la cual se empezó a hablar de seguridad en DNS, se puede decir que el tema se volvió serio después de que Steve Bellovin descubriera en 1990 serias fallas de seguridad en el protocolo. En 1995 publica un paper [9] desarrollando estas vulnerabilidades, dejando en evidencia que era indispensable comenzar a trabajar en un mecanismo de seguridad. Las primeras aproximaciones a DNSSec fueron publicadas por la IETF en 1997, la RFC 2065 [10], seguida por la RFC 2535 [11] en 1999. Sin embargo, enseguida quedó en evidencia que esta nueva extensión no era escalable en redes grandes, mucho menos en Internet, por lo que terminaron quedando obsoletas. La extensión de protocolo DNSSec original definida en esas RFCs requería un intercambio muy elevado de mensajes entre la zona padre y la hija, especialmente al momento de cambiar de claves, en la que la zona hija le debía mandar toda su información a su zona padre para que esta la firmara. Lo cual es inmanejable para grandes TLD.

En marzo del 2005, nace lo que se denominó DNSSec-bis para distinguirla de la original DNSSec. En las RFC 4033 [12], 4034 [13] y 4035 [14] se especifican los nuevos requerimientos, sus nuevos registros de recursos y los cambios en el protocolo. DNSSec−bis es la extensión que hoy conocemos, agregándole algunas facilidades más como NSEC3 y la automatización de las actualizaciones de claves de confianza. Igualmente, a no ser que se aclare lo contrario, seguiremos hablando de DNSSec para referirnos a la nueva extensión de modo de no estar sobrecargando la lectura.

Desde esa fecha, se está trabajando en el despliegue de DNSSec, esperando que algún día podamos contar con un sistema de nombres de dominio seguro. Cada administrador de zona puede optar por implementar DNSSec, si bien no es obligatorio, se espera que se genere una concientización sobre su necesidad a medida que zonas de niveles superiores se vayan firmando. El 15 de Julio del 2010 se firmó la zona raíz,

en una ceremonia celebrada con numerosos representantes de la comunidad de Internet. Ese mismo mes fue también firmada la zona edu, en diciembre la zona net y en marzo del 2011 la zona com. A estos dominios los siguieron otros gTLD de primer nivel como org, biz, museum y otros. En cuanto a los ccTLD, tenemos en América Latina a Brasil –br–, Colombia –co–, Chile –cl– que ya firmaron sus respectivas zonas. En el resto del mundo tenemos a los Países Bajos –nl– grandes promotores de DNSSec, EEUU –us–, Francia –fr–, Alemania –gr– y a varios otros más. Como curiosidad, Verisign mantiene estadísticas [15] de la cantidad de dominios firmados bajo las zonas edu, com y net. La zona com lidera el ranking con 6800 dominios firmados al día de la fecha.

Se espera que el despliegue completo –o en una buena proporción– dure unos cuantos años. Sin embargo son numerosos los esfuerzos que se están llevando a cabo en apoyo a DNSSec. El Instituto de Estándares y Tecnología de EEUU –NIST– del Departamento de Comercio, le requerirá a todas las agencias gubernamentales de ese país que validen y soliciten firmas DNSSec [16]. También se están desarrollando librerías capaces de ofrecer servicios DNSSec en el cliente, en especial cuando la comunicación entre los ISPs y el cliente final –última milla– no es segura.

3.2 - Datos actuales

3.2.1 - Ataques actuales a servidores DNS ataque de phishing

Los ataques al DNS son bastante comunes, aunque no todos generan mucho daño. En general estos ataques están asociados con el spam, los cuales más que dañar causan molestia. Sin embargo, algunos de estos ataques peligrosos terminan siendo exitosos resultando en serios problemas. Quisimos relatar uno de los ataques al DNS más relevante del 2011 que encontramos documentado en [17].

La filial brasileña de un reconocido banco fue víctima de un ingenioso ataque de phishing, llevado a cabo en el pasado año. Según Zscaler, un proveedor de seguridad que detectó el ataque, se trata de un ataque de envenenamiento de cache DNS de que resolvían la dirección del banco brasileño.

Una vez que el usuario ingresaba al sitio web, la víctima que proporcionaba sus credenciales estaba realizando el acceso a otro sitio que, por supuesto, no era la página oficial del banco.

Sin embargo, no se trataba de un ataque de phishing convencional. Para llevar a cabo el robo de información, los atacantes se aprovechaban de un mecanismo que resulta mucho más efectivo que los métodos tradicionales, ya que en los ataques por envenenamiento de cache, el delincuente no necesita enviar correos electrónicos a posibles clientes del banco esperando que alguno de ellos decida seguir el vínculo y caer en la estafa. En este caso, por lo contrario, solamente es necesario esperar que la posible víctima ingrese a la página brasileña del banco y esperar que el servidor DNS afectado re-direccione la víctima hacia el sitio infectado, sin que el usuario pueda percatarse al respecto.

3.3 - Ley SOPA y DNSSec

Uno de los temas más controversiales en los últimos meses en relación a Internet y DNSSec fue la denominada Ley SOPA. Dado el impacto que su aplicación puede llegar a tener en toda la Internet, consideramos relevante dedicar unos párrafos a describir la extensión de esta ley y su relación con DNSSec. Por más información referirse a [18].

El proyecto de ley SOPA requiere que los servidores de nombre dejen de remitir los números IP correspondientes al nombre de los sitios infractores. Es de destacar que el proyecto de ley no bloquea la dirección numérica del sitio, sino la referencia del servidor DNS, por lo tanto cualquier persona que conozca la dirección IP del sitio con el que se quiere contactar, puede hacerlo saltándose el bloqueo. Esto es lo que se conoce informalmente como el Efecto Halo de la ley SOPA, refiriéndose además a que muchos servidores podrían emigrar del territorio estadounidense, derivando el tráfico de Internet en un halo que rodearía el territorio afectado por la ley.

Este proyecto de ley fue estudiado por muchos especialistas en seguridad como Steve Crocker y Dan Kaminsky que publicaron un White Paper donde plantearon que no hay diferencia entre un servidor de nombres sujeto a un interdicto judicial y una falla causada por un servidor de nombres hackeado. También un estudio de los Laboratorios Sandia, una agencia de investigación del Departamento de Energía de los Estados Unidos, emitieron un documento donde declaran que “el filtrado de DNS es poco probable que resulte eficaz, tendría un impacto negativo sobre la seguridad de Internet y podría retrasar la plena aplicación del DNSSec.”

Han aparecido muchas preocupaciones en relación a que la ley SOPA pudiera dañar la utilidad de las Extensiones de Seguridad del Sistema de Nombres de Dominio –DNSSec–. En un White Paper emitido por la Brookings Institution puede leerse que: “El sistema DNS está basado en la confianza”, añadiendo que “el sistema DNSSec fue desarrollado para prevenir re-direccionamientos maliciosos del tráfico DNS” y que “cualquier otra forma de re-direccionamiento destruiría las garantías de esta herramienta de seguridad.”

Al día de la fecha, la aprobación de esta ley se encuentra detenida en espera de mayor consentimiento a nivel mundial. Esta ley llevo a repercusiones mundiales, como el cierre de 24 horas de muchos sitios como Wikipedia y Facebook el día 18 de Enero de este año.

4. Conceptos Importantes de DNSSec

En esta sección hablaremos de los conceptos importantes que definen a DNSSec, que luego profundizaremos a lo largo del documento. Como fue mencionado anteriormente, DNSSec proporciona autentificación de origen, integridad de los datos DNS, y mecanismos para la distribución de claves públicas.

4.1 - Definición de Terminología

En este fragmento, hablaremos de las definiciones y conceptos propios de DNSSec, los cuales se repetirán a lo largo de la documentación. En general, intentaremos traducir dichos términos, pero muchas veces el inglés expresa mucho mejor el concepto a definir, por lo cual lo mantendremos.

RRset: Heredaremos este término inglés para indicar el grupo de RR −Registros de Recursos− de un dominio que comparten el mismo nombre, clase y tipo. Cuando hagamos referencia al RRset A de ejemplo.com, estaremos agrupando todos los registros tipo A del domino ejemplo.com.

Ejemplo 4-1: RRset

```
ejemplo.com. IN     A  IP1
ejemplo.com. IN     A  IP2
```

Stub Resolver: Estas entidades, presentes en los sistemas operativos de los equipos de un usuario de Internet, son las encargadas de repetir las consultas DNS hacia los resolvers –típicamente a los de los ISP– que realizan las tareas pertinentes en su nombre. DNSSec abre el espectro a otros tipos de stub resolver en función de su comportamiento frente a las extensiones de seguridad. En primer lugar tenemos los security aware –conscientes de la seguridad– quienes reconocen estas extensiones y son capaces de distinguir entre datos validados y no validados que el resolver le envía. Un stub resolver security aware pude definirse como “validador” o “no-validador” dependiendo de si el stub resolver verifica las firmas DNSSec por su cuenta o confía en un servidor de nombres security-aware para hacerlo.

Stub Resolver Security-Aware Validador: Es un resolver security-aware que envía consultas en modo recursivo, pero realiza la validación de la firma por su cuenta en vez de tan solo confiar ciegamente en un servidor de nombres recursivo security-aware. Estas entidades juegan un papel importante en el despliegue de DNSSec, ya que deben, en última instancia, transmitir los datos a las aplicaciones de los usuarios. Se espera que los sistemas operativos incorporen este último tipo de stub resolvers.

Stub resolver Segurity-Aware No Validador: Este stub resolver confía en uno o más servidores de nombres recursivos security-aware para llevar a cabo la mayor parte de las tareas a su nombre. En particular, un resolver stub security-aware no validador es una entidad que envía consultas DNS, recibe una respuesta DNS, y es capaz de establecer un canal de seguridad adecuado para el servidor de nombres recursivo que presta estos servicios en nombre de este resolver stub security-aware.

Servidor de Nombre Security-Aware: Entidad que actúa en el rol de un servidor de nombres, este entiende las extensiones de seguridad de DNS. En particular, un servidor de nombres security-aware es una entidad que recibe consultas DNS, envía respuestas DNS, soporta la extensión de tamaño del mensaje EDNS0 y el bit DO, y soporta los tipos RR y los bits del encabezado del mensaje definidos más adelante.

Servidor recursivo Security-Aware seguridad de DNS. En particular, un security-aware resolver es una entidad que envía preguntas DNS, recibe respuestas DNS, soporta mensajes de extensión de tamaño EDNS0 y el bit DO y es capaz de utilizar los tipos RR y los bits de encabezado de mensaje definidos para proveer servicios DNSSec.

Security-Oblivious: cualquier entidad que no sea segurity-aware. Nos referiremos a este tipo de entidades cuando son transparentes a DNSSec.

Cadena de Autenticación: Diremos que estamos en presencia de una Cadena de Autenticación, cuando tenemos una secuencia alternada de RRsets de clave pública DNS –DNSKEY RR− y RRsets de Delegation Singner –DS RR− los cuales forman una cadena de datos firmados. Cada eslabón de la cadena –DNSKEY y DS− certifica al siguiente eslabón. El registro DNSKEY es utilizado para verificar la firma que cubre un DS RR, lo que permite que el DS RR sea autentificado. El DS RR contiene un hash de otro DNSKEY RR –otro eslabón de la cadena− y este nuevo DNSKEY RR es autenticado cuando coincide con el hash del DS RR. Este segundo DNSKEY RR, a su vez autentica otro DS RR, el cual contiene un hash del DNSKEY RR de otro eslabón. Finalmente, luego de sucesivas iteraciones, se termina con un DNSKEY RR cuya correspondientes clave privada firma los datos DNS deseados.

Clave de autenticación: Es una clave pública que un resolver security-aware ha verificado, y por lo tanto, se puede utilizar para autenticar los datos. Un security-aware resolver puede obtener las claves de autenticación de tres formas. Primera, el resolver es generalmente configurado para conocer por lo menos una clave pública –conocida como *Ancla de confianza*−. Esta es usualmente la propia clave pública o un hash de esta clave pública como el que se encuentra en el RR DS en la zona padre. Segunda, el resolver puede usar una clave pública ya autentificada para verificar un RR DS y el RR DNSKEY al que el RR DS se refiere –siguiendo la cadena de autenticación−. En tercer lugar, el resolver puede ser capaz de determinar que una nueva clave pública ha sido

firmada por la clave privada correspondiente a otra clave pública que el resolver ya ha verificado. Se debe tener en cuenta que los resolvers están guiados por políticas locales, las cuales pueden no permitir este tercer método.

Ancla de Confianza −Trust Anchor−: En una cadena de confianza, el ancla será su primer eslabón. Esta ancla, es un DNSKEY RR o su correspondiente DS RR y es utilizada como punto de partida para la construcción de dicha cadena. Un resolver que valide firmas deberá tener almacenada esta clave pública, o su hash, si pretende autentificar registros dentro de su isla de seguridad. En general, estos resolvers tendrán que obtener los valores iniciales del Ancla de Confianza a través de algún medio seguro y de confianza, fuera del protocolo DNS. En el caso ideal de que DNSSec esté completamente desplegado, la única ancla de confianza que necesitaremos será la del root, pero hasta entonces tendremos tantas como islas de seguridad. Una solución temporal a este problema es DLV –DNSSec Look-aside Validation− de ISC que veremos más adelante en la sección 7.3.1.2 - .

Punto de Delegación: Término utilizado para describir “el nombre” en el lado paternal de un corte de zona. Es decir, el punto de delegación para "foo.example" sería el nodo foo.example en la zona "example" −de forma opuesta se nombra *zona ápex* desde la zona “foo.example”−.

Zona Ápex: Es el nombre que se le da a la zona padre para distinguirla de las hijas en el punto de delegación o corte de zona.

RRset Autoritativos: un RRset es “autoritativo” si y solo si el propietario del nombre del RRset se encuentra dentro del espacio de nombres que está en o por debajo de la zona ápex y en o por encima de los cortes que separa la zona de sus hijos, si lo hubiera.

Isla de Seguridad: Como lo ilustra la Figura 4.1, se dirá que una zona es una Isla de Seguridad cuando está firmada pero no contiene una cadena de autentificación hacia su zona padre. Es decir, la zona padre no tiene un DS RR que contenga el hash de un DNSKEY RR de la zona hija por lo que no puede pertenecer a una cadena de confianza. Una isla de seguridad es atendida por los servidores de nombre security-aware y puede proporcionar autenticación para todas las zonas hijo delegadas. Las consultas de una isla de seguridad, o de sus descendientes, sólo pueden ser autenticadas si se obtiene su ancla de confianza por algún medio confiable fuera del protocolo DNS.

Key Signing Key –KSK−: La KSK es una clave de autenticación que corresponde a una clave privada, utilizada para firmar claves de autenticación de una zona determinada. Normalmente, la clave privada correspondiente a una KSK firmará una ZSK −Zone Signing Key−, que a su vez tiene una clave privada correspondiente que firmará los datos de la zona. DNSSec recomienda que la ZSK sea cambiada con frecuencia, mientras que la KSK puede tener un período de validez más largo, con el fin de proporcionar un punto de entrada más estable hacia la zona. Designar una clave de autenticación como una KSK es puramente una cuestión operativa, la validación de DNSSec no distingue entre las KSK y otras claves de autenticación de DNSSec, y es posible utilizar una sola clave como una KSK y una ZSK.

Zone Signing Key –ZSK–: Es una clave de autenticación que corresponde a una clave privada, que es utilizada para firmar los registros de una zona. Por lo general, un ZSK pertenecerá al mismo DNSKEY RRset al que pertenece la KSK cuya clave privada se utiliza para firmar dicho DNSKEY RRset. Como ya mencionamos, la diferencia entre la KSK y la ZSK es puramente operativa, y aunque DNSSec recomienda la diferenciación dependerá de las políticas locales.

Zona Firmada: Es una zona cuyos RRsets están firmados y esta contiene una construcción correcta del DNSKEY, firma de Registros de Recursos –RRSIG–, Next Secure –NSEC–, y los registros DS –si los hubiese–.

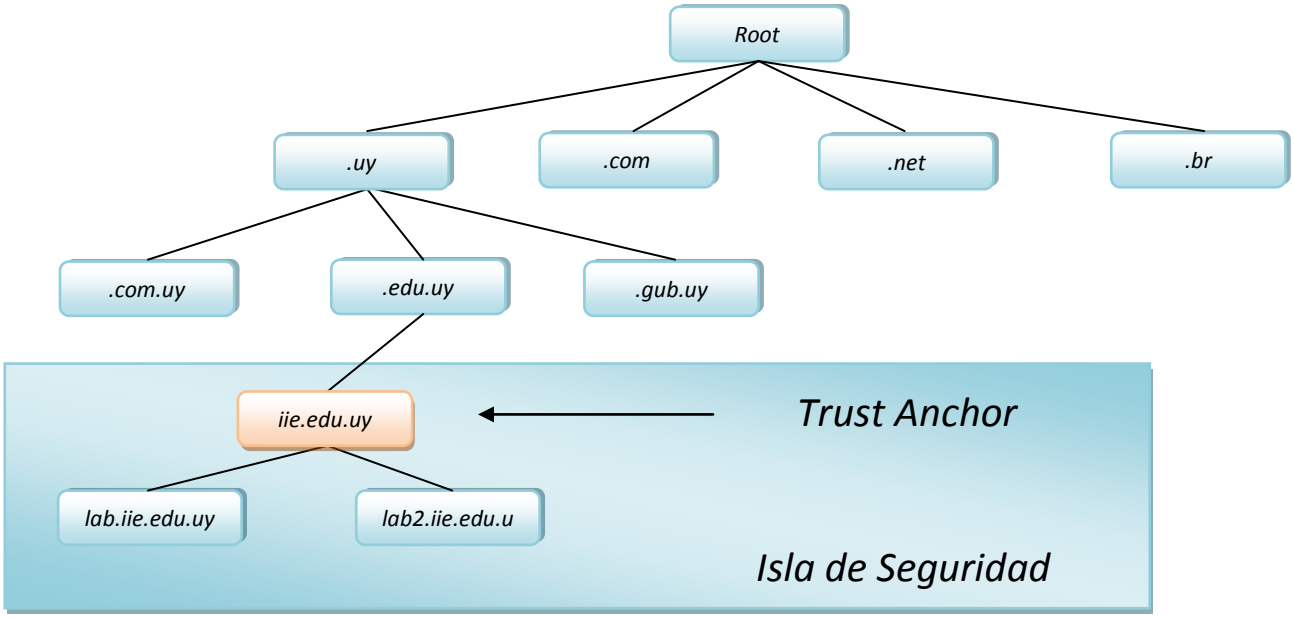


Figura 4.1: Isla de seguridad y Trust Anchor:

4.2 - Servicios que ofrece DNSSec

DNSSec ofrece los siguientes servicios entre el servidor autoritativo y los resolvers security aware.

- Autenticación de origen
- Integridad de datos DNS
- Autenticación de negación de existencia de datos DNS.

El protocolo DNS requirió de un cambio para incorporar estos servicios. DNSSec añade cuatro nuevos registros de recursos:

- Resource Record Signature: RRSIG

- DNS Public Key: DNSKEY
- Delegation Signer: DS
- Next Secure: NSEC y NSEC3

También se redefinen dos bits reservados en el encabezado de mensaje:

- Checking Disabled: CD
- Authenticated Data: AD

Con el fin de soportar los mensajes de mayor tamaño del DNS que resultan de agregar estos nuevos registros, los servidores de nombre que soporten DNSSec también deben soportar EDNS0 y el bit DNSSec OK –DO− del encabezado EDNS, con el cual un resolver security-aware pueda indicar en sus consultas si desea recibir registros DNSSec en los mensajes de respuesta.

4.3 - Servicios no proporcionados por DNSSec

Originalmente DNS fue diseñado con la intención de que todas las respuestas devueltas sean independientes de quien haya emitido la consulta, y por lo tanto todos los datos DNS serían visibles y públicos. En consecuencia a esto, DNSSec no está diseñado para garantizar confidencialidad, listar controles de acceso, o cualquier otro medio que diferencie a los indagadores. El protocolo de ninguna manera restringe el acceso de usuarios a datos DNS, si no que ofrece mecanismos para autenticar los mismos. Uno de los aspectos más importantes en el despliegue de DNSSec es su compatibilidad hacía atrás, lo que determina que servidores de nombre que no reconocen DNSSec puedan seguir operando y funcionando normalmente en presencia de datos firmados.

DNSSec no proporciona protección contra ataques de negación de servicio −DoS−. Los resolvers security-aware y servidores de nombres security-aware son vulnerables a una clase adicional de ataques de este tipo, ya que las operaciones criptográficas que se realizan en estos los hace más vulnerables a ataques de DoS.

Las extensiones de seguridad DNS proporcionan autenticación de datos y origen de los datos DNS. Los mecanismos descritos anteriormente no han sido diseñados para proteger las operaciones como las transferencias de zona y actualizaciones dinámicas. Estas siguen operando bajo los estándares de seguridad ya establecidos, y son complementarios a DNSSec.

5. Autenticación de origen e integridad de datos

Dedicaremos ahora nuestra atención a explicar de forma generalizada las entidades y procesos que llevan a cabo la autenticación de origen e integridad de datos en DNS.

5.1 - Cadena de Confianza

DNSSec ofrece autenticación mediante la asociación criptográfica de firmas digitales de los RRsets, denominada *cadena de confianza*. Esta cadena comienza con lo que se denomina trust anchor, ancla de confianza. Esta ancla es una clave pública que se obtuvo y verificó por medios externos a DNS y de la cual se tiene certeza de su integridad y autenticidad. El ancla de confianza por excelencia es la clave pública de la zona raíz, la cual está firmada desde Julio del 2010. Esta se puede obtener también por medios externos a DNS [22] aunque los servidores de nombre como BIND ya traen configurada esta clave. A la zona cuya clave utilizamos como ancla de confianza se denomina zona SEP −Secure Entry Point−. Cualquier zona puede ser considerada una zona SEP, siempre que tengamos una clave de confianza configurada como tal en nuestro servidor de nombre. En caso de que la zona SEP no sea la zona raíz, estamos en presencia de una *isla de seguridad*, que veremos más adelante.

Uno de los principios de DNSSec es lograr transferir la confianza ya depositada en el ancla de confianza hacia claves de zonas inferiores, de modo de poder también validar sus datos. El propósito por lo tanto de la cadena de confianza es la transferencia de esta confianza de forma segura y confiable.

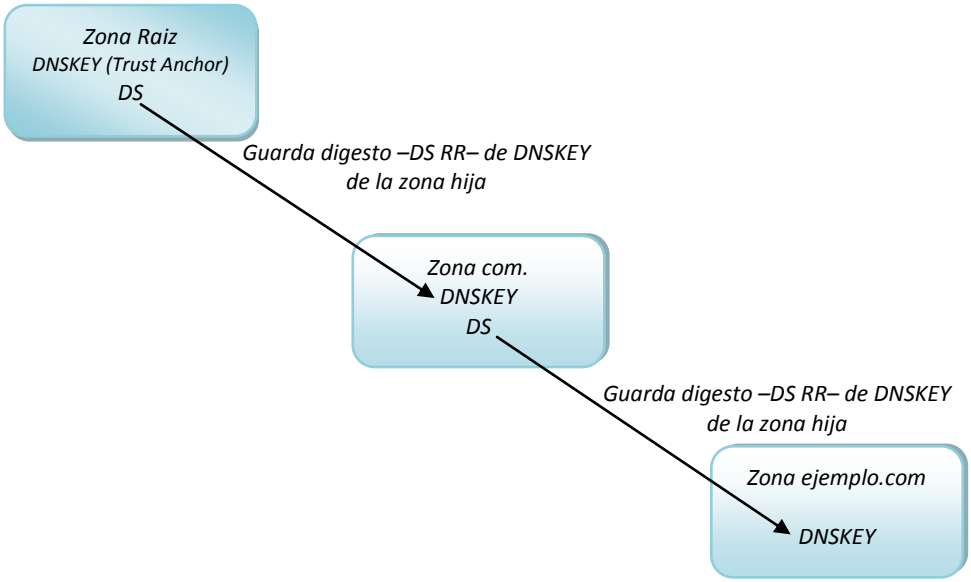


Figura 5.1: Cadena de autenticación

Supongamos el escenario de la Figura 5.1, donde inicialmente tenemos configurada la zona raíz como nuestra zona SEP, por lo que tenemos a disposición la clave pública de dicha zona, y queremos validar datos de la zona com. Para lograr esto, DNSSec introdujo el registro DS −Delegation Signer−, el cual contiene un hash de la clave pública de la zona hija, en este caso com. En la zona padre −raíz− deberá estar presente el RR DS perteneciente a com, y cuya integridad se puede verificar validando el RRSIG del registro DS. Por otra parte, la zona com −la cual deberá estar firmada por supuesto−, tendrá presente un RR DNSKEY conteniendo la clave pública KSK de com. Con este DNSKEY y su respectivo DS, se puede verificar si la parte pública de la KSK de com que obtuvimos es auténtica o no. Paso por paso:

1. Obtener el RR DS de com en la zona raíz.
2. Verificar la integridad y autenticidad del RR DS de com, utilizando la clave pública de la zona raíz que ya tenemos.
3. Obtener el RR DNSKEY de la KSK de la zona com, en su propia zona.
4. Realizar el hash de este último DNSKEY: H1(KSK).
5. Comparar H1(KSK)==DS(com).

Si la comparación resulta positiva, como ya verificamos el RR DS, sigue inmediatamente que la clave pública KSK de com es auténtica y puede utilizarse para validar firmas dentro de la zona com., en particular validar las firmas de la ZSK. Si de lo contrario la comparación resulta negativa, entonces no es posible validar la clave pública de com.

Asumiendo un escenario positivo, logramos obtener la trasferencia de confianza que necesitábamos. En este punto podemos hacer exactamente lo mismo

con zonas inferiores, que estén firmadas. La cadena de confianza está formada entonces por esta asociación criptográfica del registro DS en la zona padre y su respectivo registro DNSKEY en el ápex de la zona hija. Un concepto importante de DNSSec es que la clave que firma los datos de una zona es asociada a la propia zona y no con los servidores de nombres autoritativos para la zona, de los cuales pueden existir muchos.

5.2 - Autenticación de datos

Una vez recorrida la cadena de confianza hasta la zona de donde obtenemos los registros que nos interesan, y verificada cada iteración de esta cadena, podemos asumir que la clave pública KSK de esta zona es auténtica. Con esta clave tendremos que verificar la ZSK que se utilizará para verificar los datos DNS.

En primer lugar, como veremos más adelante, la clave KSK se utiliza para firmar claves, en particular la ZSK. Esta última es la que se utiliza para firmar los registros de toda la zona, y es la que en verdad nos interesa validar. Al igual que la KSK, la ZSK esta almacenada en un RR DNSKEY en el ápex de la zona, por lo que obtenerla no es el problema, si no autenticarla. Esto se realiza en un esquema conocido de la encriptación de clave pública: verificación de firmas digitales.

Estas firmas digitales son almacenadas en un nuevo registro de recurso, el RR RRSIG. Por lo general, hay una sola clave privada que firma los datos de una zona, pero es posible el uso de múltiples claves simultáneas para firmar registros. Si un resolver security-aware aprende confiablemente –veremos más delante de que se trata esto– una nueva clave pública de una zona, puede autenticar los datos firmados de esa zona con esta nueva clave.

Una vez obtenida la clave, el resolver será capaz de autenticar los datos de la zona verificando las firmas digitales, de la siguiente manera:

1. Se obtiene el RRset cuya integridad y autenticidad se quiere verificar.
2. Se realiza el hash correspondiente a dicho RRset: H(RRset[2])
3. El resolver des-encripta la firma digital RRSIG correspondiente a dicho RRset, D_{ZSK}(RRSIG).
4. Si la comparación D_{ZSK}(RRSIG) == H(RRset) da correcta, se puede asegurar la autenticidad e integridad de los datos obtenidos.

El punto 4 es consecuencia de las propiedades de la encriptación asimétrica y las funciones de hash utilizadas en criptografía. Los algoritmos de hashes deben asegurar dos cosas, primero que el resultado no pueda ser “revertido”, obtener el texto plano a partir de su hash. Segundo, el algoritmo no debe permitir que se pueda

[2] Veremos en el Anexo C la sección **El campo Firma** que además de utilizar el RRset también se utiliza el encabezado del propio RRSIG.

forzar que dos entradas diferentes tengan hashes iguales, lo que se denomina colisiones de hash. Los hashes son importantes ya que reducen la cantidad de texto plano que se encripta, siendo esta última operación la que más CPU consume. Las propiedades de encriptación asimétrica aseguran que los datos encriptados con la clave privada solo pueden ser des-encriptados por su respectiva clave pública. Este mecanismo asegura entonces tanto la autenticidad como la integridad de los datos. Como se menciono en el punto 3 del algoritmo de validación de firma, no solo el RRset se utiliza para generar la firma digital, sino que también es necesario incluir otros datos de no menor importancia. Además de los datos DNS, también deben autenticarse información adicional almacenada en el campo RDATA del propio RRSIG.

Es importante que un resolver validador pueda verificar la autenticad del periodo de validez de la firma presente en el encabezado del RRSIG. Si estas no son firmadas un atacante podría modificarlas con el objetivo de caducar antes de fecha las firmas, sin que nadie se percate del hecho. También deben estar incluidos en la firma digital los campos del RRSIG que indican el tipo de algoritmo utilizado en su generación, RRset que cubren y ápex de la zona entre otros datos.

En el Anexo C se especifica cómo se realiza una firma digital, en particular, como se ordena el RRset de forma inequívoca y como se incluye el RDATA del RRSIG en la firma. Se debe notar que en el RDATA del RRSIG se almacena la propia firma, la cual debe ser excluida de los datos que se usan para firmar.

El procedimiento de validación permite verificar cualquier firma digital presente en la zona. Observar que este mismo mecanismo es el utilizado para verificar la autenticidad de los registros DS al recorrer la cadena de confianza, y de ninguna forma la verificación de los mismos difiere del de los otros RRset de una zona.

Es necesario entonces que todos los resolvers tengan por lo menos un trust anchor configurado –como la de la zona raíz−, y que puedan establecer una cadena de confianza hasta el nodo final. Como ya mencionamos, también se puede iniciar una cadena de confianza desde otro punto cualquiera del árbol DNS. Esto generalmente sucede cuando la zona hija está firmada pero su zona padre no lo está. En este contexto un resolver puede utilizar a esta zona hija como SEP, y validar todas las zonas por debajo de la misma. A esto se le llama isla de seguridad. En las primeras etapas del despliegue de DNSSec, es de esperar la presencia de numerosas islas de seguridad, lo que requiere por lo tanto la configuración de la misma cantidad de anclas de confianza.

5.3 - Autenticación de no existencia de nombre y tipo

Otro cambio que introdujo DNSSec fueron los mecanismos de autenticación de no existencia. El protocolo permite dos versiones para este tipo de autenticaciones, NSEC y NSEC3. Sus diferencias y funcionamientos los veremos de forma detallada en el capítulo 0La principal dificultad que presenta la autenticación de no existencia, es que no es posible tener registros firmados para todo tipo de consulta inexistente. Tampoco

era posible considerar la realización de firmas a demanda cada vez que llegara una consulta por un nombre o tipo inexistente. Se debe considerar que en 1999 cuando comenzaba a definirse DNSSec, la capacidad de procesamiento de las computadoras no permitía ese tipo de lujos, por lo que DNSSec fue ideado con la idea de que todas las firmas necesarias ya deberían estar presentes en el servidor durante la operación de una zona. Con esto en mente, surgió NSEC, el cual encadena los nombres existentes dentro de la zona, especificando así de forma implícita los nombres que no existen. Como veremos en la sección 6.2.2 - esto trajo consigo un problema denominado enumeración de zona, el cual permite de forma muy sencilla que usuarios ajenos a la administración de la zona puedan obtener el contenido de la zona. La discusión sobre si esto era realmente o no un problema, terminó desembocando en lo que se denominó NSEC3, el cual es una alternativa a NSEC.

5.4 - Determinación del estado de los datos

Hasta ahora se menciono que un resolver validador puede determinar si los datos son seguros o no. Sin embargo, existen estados intermedios los cuales son más informativos que la discriminación entre seguro o no. Un resolver validador deberá poder inequívocamente distinguir 4 estados posibles de los datos DNS que valida −o no−. Se describen a continuación las hipótesis necesarias para asumir alguno de estos estados.

Seguro: El resolver validador tiene un ancla de confianza, una cadena de confianza completa y es capaz de verificar todas las firmas en la respuesta.

Inseguro: El resolver validador tiene un ancla de confianza y una cadena de confianza incompleta. Esto es que el resolver en algún punto de delegación, recibe una prueba de no existencia de un registro DS necesario para continuar la cadena de confianza. Esto indica que las ramas posteriores en el árbol no pueden ser validadas con dicha cadena de confianza. Sin embargo, la zona final sí puede estar firmada y pertenecer a una Isla de seguridad, pero de todos modos el resolver no puede validar los datos. En estas condiciones los datos obtenidos deben considerarse inseguros, aunque depende de una política local si utilizarlos o no.

Falso −Bogus−: El resolver validador tiene un trust anchor y una cadena de confianza completa, pero la respuesta no puede validarse por alguna otra razón: perdida de firmas, expiración de la firma, algoritmos no soportados por el servidor, falta de datos relevantes en el RR NSEC que debe estar presente, y así sucesivamente. En estos casos se debe asumir que los datos pudieron haber sido falsificados, y de ninguna manera deben utilizarse los datos obtenidos.

Indeterminado: No hay un ancla de confianza que indique que una parte específica del árbol es segura.

Los servidores de nombres security-aware pueden señalar a los stub resolvers no validadores que los datos fueron encontrados bogus −falsos− utilizando RCODE 2, "SERVFAIL", según se define en la RFC 4035.

Los servidores de nombres security-aware deben señalar a los stub-resolves security-aware que los datos fueron encontrados seguros utilizando el bit AD –Authenticated Data−. Tener en cuenta que el stub-resolver no debería confiar en este bit a no ser que el canal entre ambas entidades sea seguro, ya que DNSSec no asegura el encabezado DNS.

5.5 - Consideraciones

Existen ciertas consideraciones que las zonas y servidores de nombre deben tener en cuenta al utilizar DNSSec.

5.5.1 - Resolver

Los resolvers deben considerar ciertos aspectos que determinan que DNSSec funcione correctamente. Un resolver security-aware debe ser capaz de realizar las funciones criptografías necesarias para la verificación de firmas. Estos también deben ser capaces de formar una cadena de autenticación desde una zona SEP, como se explicó anteriormente. Un security-aware resolver debe ser por lo tanto configurado con al menos un ancla de confianza como punto de partida, desde la cual intentará establecer las cadenas de confianza. También se exige que estos servidores recursivos sean capaces de soportar hasta 5 claves de confianza por SEP.

Si un resolver security-aware está separado de los servidores de nombres autoritativos relevantes, por medio de un servidor de nombre recursivo que actúe como proxy DNS −ver Anexo D−, el resolver security-aware será incapaz de operar en modo seguro si este proxy no reconoce DNSSec.

Del mismo modo, si un resolver security-aware debe basarse en una zona sin firmar o un servidor de nombres que no es security-aware, el resolver no será capaz de validar las respuestas DNS y se necesitará una política local sobre si se acepta respuestas no verificadas o no.

Un resolver security-aware debería tomar en consideración el período de validación de una firma para determinar el TTL de los datos de su cache, para evitar el almacenamiento en cache de firmas después del período de validez de las mismas. Sin embargo, también se debe tener en cuenta la posibilidad de que el reloj del propio resolver security-aware este mal. Por lo tanto un resolver security-aware que es parte de un servidor de nombres recursivo security-aware tendrá que prestar especial atención al bit “checking disabled” −CD− de DNSSec. Esto es con el fin de evitar el bloqueo de firmas válidas obtenidas a través de otros security-aware que son clientes de este servidor de nombres recursivo.

5.5.2 - Stub resolver.

Los stub resolvers deben cumplir también con ciertas consideraciones −dependiendo si es de tipo validadores o no validadores− para mantener la integridad de los datos y la distribución de las claves.

Incluso un stub resolver security-oblivious puede beneficiarse de DNSSec si el servidor de nombres recursivo que es utilizado es security-aware. Sin embargo, para que el stub resolver coloque cualquier confianza real sobre estos resolvers, el stub resolver debe confiar tanto en el servidor de nombres recursivo en cuestión, como en el canal de comunicación que lo comunica con él. El confiar en el servidor recursivo es una cuestión de política local: un stub resolver security-oblivious no tiene más opción que ponerse a merced de los servidores de nombres recursivos que utiliza, ya que no realiza comprobaciones de validez de DNSSec por sí mismo; y el confiar en el canal de comunicación requiere de algún tipo de mecanismo de canal de seguridad; el uso apropiado de los mecanismos de autenticación en transacciones DNS como SIG(0) o TSIG serían suficientes, al igual que el uso apropiado de IPSec. La confidencialidad no es necesaria para este canal, pero la integridad de datos y autenticación de mensajes si lo son.

Un stub-resolver security-aware que confía tanto en sus servidores de nombres recursivos como en su canal de comunicación, puede elegir examinar el setting del bit AD −Authenticated Data− en el encabezado del mensaje de respuesta que recibe. El stub resolver puede usar estas flag para saber si el servidor de nombres recursivo fue capaz de validar las firmas de todos los datos en las secciones Answer y Authority de la respuesta.

Hay un paso más que un security-aware stub resolver puede tomar, si por cualquier razón, no es capaz de establecer un canal seguro con los servidores de nombre recursivo que utiliza: este puede realizar su propia validación de firma haciendo un setting en el bit CD en el mensaje de consulta. Un stub resolver validador será capaz de manejar firmas DNSSec y establecer una relaciones de confianza entre los administradores de zona y el propio stub resolver.

5.5.3 - Zonas

Una zona firmada contendrá adicionalmente los registros relacionados con la seguridad −RRSIG, DNSKEY, DS y los registros NSEC o NSEC3−. A los registros RRSIG que escoltan los datos de zona se les definen tiempos de inicio y fin, los que establecen el período de validez de las firmas.

5.5.3.1 - Valor de TTL vs. Periodo de Valides RRSIG

Es importante tener en cuenta la diferencia entre el valor TTL de un RRset y el período de validez de la firma especificado por el RR RRSIG que cubre el RRset. DNSSec no cambia la definición o la función del valor TTL, que tiene como objetivo mantener la

coherencia de la base de datos en los caches. Una resolver purga del cache los RRsets al finalizar el período de tiempo especificado por su campo TTL, independientemente de si el resolver es security-aware o no.

Los campos de comienzo y fin en el RR RRSIG, por el contrario, especifican el período de tiempo durante el cual la firma puede ser utilizada para validar el RRset cubierto. Las firmas asociadas a los datos firmados de la zona sólo son válidas para el período de tiempo especificado por estos campos del RR RRSIG en cuestión. Los valores TTL no pueden ser mayores al período de validez de los RRsets firmados en el cache de los resolvers. El resolver puede utilizar el tiempo restante antes de la expiración de la firma de un RRset, como un límite superior para el TTL para los propios RRset y sus RR RRSIG asociados guardados en el cache del resolver.

5.5.3.2 - Nuevos problemas de dependencia temporal para las zonas

La información en una zona firmada tiene una dependencia temporal que no existía en el protocolo original DNS. Una zona firmada requiere un mantenimiento habitual donde se asegure que cada RRset en la zona tenga un RR RRSIG actual y válido. Sin embargo, las firmas de una zona pueden vencerse en tiempos diferentes, por lo que volver a firmar una o más RRsets en una zona requerirá incrementar el número de serie del SOA de la zona para indicar que un cambio de zona ha ocurrido. Por lo tanto, volver a firmar cualquier RRset en una zona también puede desencadenar en mensajes DNS NOTIFY y operaciones de transferencia de zona. Lo que significará un incremento de tráfico.

5.5.4 - Servidores de Nombres autoritativos

Un servidor de nombres security-aware debe incluir los registros apropiados DNSSec −RRSIG, DNSKEY, DS, y NSEC− en todas las respuestas a las consultas de los resolvers que han manifestado su voluntad de recibir tales registros. Esto lo hacen a través del uso del bit DO en la cabecera de EDNS, cuyo uso está sujeto a las limitaciones de tamaño del mensaje que el servidor pueda soportar. La inclusión de estos registros de recursos DNSSec puede causar fácilmente el truncamiento de mensaje UDP y fallback para TCP, por lo tanto los servidor de nombres security-aware también debe soportar el mecanismo EDNS "senders UDP payload".

La parte privada de cada par de claves DNSSec debe mantenerse offline, pero esto no será posible para una zona que tiene habilitada las actualizaciones dinámicas de DNS. Cuando se tiene configurado la actualización dinámica, el servidor maestro para la zona tendrá que volver a firmar la zona cuando esta se actualice, por lo que la clave privada correspondiente a la ZSK tendrá que mantenerse online. Este es un ejemplo en donde separar las funciones de la ZSK y KSK puede ser útil, en este caso las KSK todavía se pueden mantener offline.

Por sí mismo, DNSSec no es suficiente para proteger la integridad de toda una zona durante las operaciones de transferencia de zona. Por lo tanto, las operaciones

de mantenimiento de la zona requieren algunos mecanismos adicionales –lo más probable algún tipo de canal de seguridad, tales como TSIG, SIG (0), o IPSec–.

5.5.5 - Seguridad

Para que un resolver security-aware valide una respuesta DNS, todas las zonas a lo largo del camino desde el punto de partida confiable –SEP– hasta la zona que contiene los datos de la respuesta deben estar firmadas y todos los servidores de nombres y resolver que participan en el proceso de resolución deben ser security-aware. Si hay una ruptura en la cadena de autenticación, como por ejemplo si un security-aware resolver no puede obtener y validar las claves de autenticación que necesita, entonces el resolver security-aware no puede validar los datos DNS que necesita resolver.

Un stub resolver security-aware no validador, por definición no realiza por su propia cuanta validaciones de firma DNSSec y por lo tanto es vulnerable tanto a los ataques a los servidores de nombres recursivos security-aware como a los ataques al canal de comunicación con los resolvers. Los stub resolvers security-aware no validadores debe usar algún tipo de canal de seguridad para defenderse de estas amenazas. Si de lo contrario no se cuenta con un canal de seguridad entre el servidor recursivo y el stub-resolver, éste último será vulnerable a ataques DNS a no ser que realice su propia validación de la firma, momento en el que, por definición dejaría de ser un stub resolver security-aware no validador.

Como vimos en 2.3.2 - , DNSSec no protege contra ataques de negación de servicio, sino que los amplifica. Este tipo de ataques tiene por lo menos dos formas:

- Un atacante puede ser capaz de consumir recursos de un resolver security-aware durante la validación de las firmas, manipulando el RR RRSIG en los mensajes de respuesta o mediante la construcción de una cadenas de firmas innecesariamente compleja.
- Un intruso podría ser capaz de consumir recursos en un servidor de nombres security-aware que soporta actualización dinámica de DNS, mediante el envío de un flujo de mensajes de actualización que fuerce al servidor de nombres security-aware a tener que volver a firmar algunos RRsets en la zona con más frecuencia de lo que sería necesario.

Debido a opciones de diseño, DNSSec no proporciona confidencialidad, por lo tanto se introduce la posibilidad de que un usuario mal intencionado enumere todos los nombres en una zona, siguiendo la cadena de NSEC. El RR NSEC declara de forma implícita que nombres no existen en una zona, por medio del orden canónico de todos los nombres dentro de una zona. Por lo tanto, un atacante puede consultar estos RR NSEC en secuencia para obtener todos los nombres de una zona. Aunque esto no es un ataque al propio DNS, podría permitir a un atacante mapear los host de la red u otros recursos mediante la enumeración del contenido de una zona. Para impedir esto se creó el registro NSEC3, que también provee la negación de la existencia autenticada. Sin embargo, NSEC3 introduce otros problemas de seguridad relacionados a los

ataques de negación de servicio. Dada las operaciones criptográficas adicionales que se deben realizar, tanto en los servidores de nombre autoritativos como en los recursivos, un atacante puede saturar a un servidor con consultas que generen respuestas NXDOMAIN de forma más rápida que si se utiliza NSEC.

5.6 - Registros de Recursos DNSSec

Para determinar la autenticidad de los datos DNS la extensión del protocolo hace uso de 4 nuevos tipos de registros. Estos tienen el objetivo de almacenar la información relevante para la validación de los datos, pero de ninguna otra manera se diferencian de otros registros DNS, los cuales son públicos y pueden ser obtenidos por cualquier cliente. Se resumirá aquí como se conforma cada uno de estos nuevos registros, especificando el contenido de sus campos y sus funciones. En el Anexo C se puede encontrar información más detallada acerca de los mismos.

El RR DNSKEY almacena claves públicas y parámetros de las mismas. Estos registros pertenecen al ápex de la zona y son autoritativos, por lo que deberán tener una firma asociada a su RRset.

Ejemplo 5-1: RR DNSKEY

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3... )
```

En el ejemplo anterior se eliminó parte del contenido del registro codificado en Base64. En el RDATA de este registro se especifica que la clave que éste almacena es una ZSK −flag 256−, los algoritmos utilizados para la generación de la clave −el 5 indica RSA/SHA1, ver Tabla 7-1− y por último la clave pública propiamente dicha.

Las firmas digitales de cada RRset autoritativo de una zona se almacenan en el nuevo registro RRSIG. Cada RRset −identificado por su nombre, tipo y clase− debe tener asociado un RRSIG para que la zona se considere firmada, entre otras cosas. El RRSIG se identifica por su nombre de dominio −host.example.com en este caso− y el tipo de RRset que la firma cubre, en este caso el RRset A de dicho nombre de dominio. Además de la firma digital, en el RDATA del registro también está indicado el algoritmo utilizado en la generación de la firma −el cual debe coincidir con el algoritmo indicado en su clave pública correspondiente−, período de validez de la firma, zona a la que pertenece y otros parámetros más. La construcción de la firma no solo incluye el RRset que esta cubre, sino que también los otros parámetros contenidos en el RDATA −excluyendo a la propia firma−. Esto es necesario para garantizar también la propia integridad de la misma.

Ejemplo 5-2: RR RRSIG

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                20030220173103 2642 example.com.
                                oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr...)
```

Los registros de recursos NSEC y NSEC3 son utilizados en la autenticidad de no existencia en DNSSec. En el Anexo C se especifica el contenido y construcción de los

mismos, y en el siguiente capítulo se describirá con detalle como son los mecanismos de autenticidad de no existencia con los dos tipos de registros permitidos en DNSSec.

Por último, el registro de recurso DS −Delegation Signer− tiene la particularidad de estar contenido en la zona padre. El RR DS es quien encadena la autenticidad de los datos de la zona padre con los datos de la zona hija, a través del corte de zona.

Ejemplo 5-3: RR DS

```
dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A
                                           98631FAD1A292118)
```

El registro contiene un hash de una clave pública KSK de la zona delegada, el cual es utilizado para validar el RR DNSKEY en el ápex de dicha zona. Además de esta información, también está almacenado el tipo de algoritmo −5− utilizado para la generación de la clave, el cual debe ser el mismo que el indicado en el respectivo DNSKEY. El keytag de una clave −60485 en este ejemplo− se incluye en el RDATA del RR DS para poder identificar el DS correspondiente al RR DNSKEY cuya integridad se quiere verificar. Dado que puede existir más de un DS con el mismo nombre, es necesario poder diferenciarlos con el objetivo de no realizar operaciones criptográficas de más durante el proceso de construcción de la cadena de confianza.

5.7 - Consultas y respuestas con DNSSec

Habiendo visto los mecanismos de DNSSec para validar la integridad y autenticidad de las respuestas, veremos ahora unos ejemplos de consultas y respuestas.

Utilizando la herramienta dig de BIND, consultaremos primeramente por el RR A de example.com, que como ya se mencionó este nombre está reservado con el propocito de realizar ejemplos.

Ejemplo 5-4: Dig ejemplo.com

```
; <<>> DiG 9.8.1-P1 <<>> example.com @localhost +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39259
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;example.com.             IN     A

;; ANSWER SECTION:
example.com.        172771 IN     A      192.0.43.10
example.com.        172771 IN     RRSIG  A    8    2    172800    20120618095132
20120610212715                     29449                          example.com.
Vxxy2h9fInOxmjFYCaANOfHV8OF7ROLXWGykw9n6WlJ6Mapg7x5UoOUI
7h3qyKKA4Ytzq80+Lin1Rp58G+iNkumd4LU4/84t5h+ANBqNgw1cHt2W
caoDtWYq5GIkVFHzQnWt+UKuXUQmknFYUViPaDPsZgrcsjYQKJFee8bz YNA=

;; AUTHORITY SECTION:
example.com.        172770 IN     NS     b.iana-servers.net.
```

```
example.com.        172770 IN     NS     a.iana-servers.net.
example.com.        172770 IN     RRSIG  NS    8    2    172800    20120618051008
20120610212715                       29449                          example.com.
JDec5W62aLEr2jc39wl7j03ev0jB+Ur4AiwOBinwivcrXAsRPwmhqqou
mG5uFwekv1ZAa3MOM9c+pRKDeIsWJJyQ0/kHyavPIba0y1Tk6U1ft5od
lSETkqlfhroV5xZ8x+urTok7KLGPVErChG98NLKIovxxja1glgZznssg 0KE=

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Jun 10 22:25:47 2012
;; MSG SIZE  rcvd: 446
```

En este ejemplo se utilizó el servidor de nombres recursivo local –localhost–, el cual está configurado con el ancla de confianza de la raíz, lo que nos permitirá validar los datos de todos las zonas firmadas que puedan establecer una cadena de confianza hasta la raíz.

Los primeros elementos de la respuesta a tener en cuenta son el encabezado DNS y el OPT Pseudosection de EDNS. En el encabezado tenemos un código de respuesta –RCODE– NOERROR y las flags que se encuentran seteadas en la respuesta. Observar en estas que la flag AD –Authenticated Data– indica que los datos fueron correctamente validados por nuestro servidor local, por lo que podemos tener confianza de que la respuesta es autentica. Si el servidor utilizado en la validación fuera uno externo, como el de las ISP, solo podremos confiar en este bit si confiamos en el canal de comunicación entre nuestro equió y el servidor de nuestra ISP. Si dicho canal no es considerado seguro, el cliente debe solicitar las firmas digitales y realizar la validación por sí mismo.

El servidor autoritativo de example.com. debe enviar toda la información necesaria para la validación junto con la respuesta –firmas digitales–. Al momento de armar el paquete con la respuesta, primero introduce los datos DNS los cuales en este caso es el RR A de example.com., luego debe introducir la firma digital que cubre la respuesta. Si dicho RRSIG no entrase en el paquete –cuyo tamaño máximo fue declarado en el OPT pseudosection de EDNS– se debe setear el bit TR –truncated– para indicar que la respuesta no está completa. Una vez que nuestro servidor validador local tiene los datos y la firmas digitales, realiza la validación correspondiente, utilizando los datos provistos en el RDATA del RRSIG.

Supongamos ahora que realizamos una consulta por un nombre no existente en la zona example.com. Consultaremos por noexiste.example.com. y analizaremos lo que recibimos.

Ejemplo 5-5: Dig noexiste.example.com.

```
; <<>> DiG 9.8.1-P1 <<>> noexiste.example.com. ANY @localhost +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 47376
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;noexiste.example.com.           IN     ANY
```

```
;; AUTHORITY SECTION:
example.com.        3600   IN     SOA    dns1.icann.org.    hostmaster.icann.org.
2011063812 7200 3600 1209600 3600
example.com.        3600   IN     RRSIG  SOA    8    2    3600    20120617223630
20120610212715                       29449                          example.com.
hafDGxmzNY1P1Lmp4MS+D/NllA5kYUtBZrWODpZs1nkYs5epqMTnpidH
G9lMYvbpIOiJf/38SB80iJq9y0Z+QuTLXSUUB0WfmzPhv1JqO7vfnhHS
zQESVuYpw+zlf8LV68A8LklnIWZdAgTXQoyzKCGjaLDLpQmdy+QdOFnh rPg=
example.com.        3600   IN     RRSIG  NSEC    8    2    3600    20120618011219
20120610212715                       29449                          example.com.
O7bugM4+V4jkPML+Mdt7u0AWMv3Cro9ILC5tWQWB1WKBWbigieI9UpQW
TI5I+7BZXzqQXbl0NQY+20ZS1XYtfsLr6xtWpV46B8wjLFrOBOynk7we
1563AmSNgXnMDroKvtI0P2vCNwa+Qds1VARrEZhAZ7HWC1PwrTGTqZL8 J8A=
example.com.        3600   IN     NSEC   www.example.com.  A  NS  SOA  AAAA  RRSIG
NSEC DNSKEY

;; Query time: 146 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Jun 10 22:29:12 2012
;; MSG SIZE  rcvd: 490
```

En este ejemplo tenemos una respuesta del tipo NXDOMAIN, la cual se puede identificar por su código de respuesta y el registro SOA en la sección autoritativa de la respuesta. Nuevamente en el encabezado tenemos el bit AD indicando que la validación de los datos recibidos fue exitosa. Analizando los registros recibidos, en primer lugar encontramos el RR NSEC que cubre el dominio consultado, veremos en el próximo capitulo el funcionamiento de los mecanismos de autenticación de no existencia en DNSSec.

6. NSEC y NSEC3

Ya vimos como DNSSec utiliza las firmas digitales para validar respuestas de dominios que existen, ahora veremos los mecanismos definidos para demostrar cuando los mismos no existen. Esto se denomina autenticidad de prueba de no existencia.

6.1 - Prueba de no existencia

En primer lugar diremos qué significa la no existencia de un dominio y luego veremos cómo probarla. En la RFC 2308 [19] se refinan los conceptos de respuestas no existentes y se dan los delineamientos para almacenar respuestas negativas en los caches. Rescataremos entonces de esta RFC las definiciones de NXDOMAIN y NODATA.

Una respuesta cuyo encabezado trae un RCODE = NXDOMAIN –anteriormente conocido como NAME ERROR en la RFC 1035− significa que el nombre de dominio solicitado no existe. Esta información por lo tanto solo la puede ofrecer un servidor de nombres autoritativo para ese dominio. El segundo tipo de no existencia es el NODATA, que a diferencia del anterior no está definido en el encabezado, si no que tiene que inferirse a partir de la respuesta del servidor autoritativo. Esta respuesta se origina cuando el nombre de dominio consultado sí existe, pero no tiene asociado el RR requerido. Se reconoce normalmente este tipo de no existencia por tener un RCODE = NOERROR y el registro SOA en la sección autoritativa de la respuesta.

Las pruebas de no existencia deben considerar tres puntos. En primer lugar se debe tener en cuenta que el protocolo no tiene definido mecanismos para firmar respuestas según las necesite, o sea, la prueba de no existencia debe ya estar firmada en el archivo de zona. Segundo, el encabezado de un paquete DNS no se firma, por lo que tanto el código de la respuesta –NXDOMAIN en este caso− como el bit AD no pueden ser considerados confiables para validar una prueba de no existencia. Tercero y último, se debe tomar en cuenta la posibilidad de que existan wildcards en la zona, según se definen en la RFC 4592 [7].

La primera iniciativa que contempló estos tres puntos mencionados fue llamado NXT, definido en la RFC 2535 [11], junto con la primera "versión" de DNSSec en 1999 ya obsoleta. Tiempo después varios nombres en DNSSec cambiaron –RFC 3755– y el NXT paso a llamarse NSEC y es uno de los mecanismos permitido por el protocolo para validar pruebas de no existencia tanto del tipo NXDOMAIN como las NODATA.

6.2 - NSEC

El funcionamiento de NSEC es bastante sencillo, en comparación con NSEC3 que veremos más adelante. Este registro es utilizado para declarar los intervalos de nombres válidos en la zona, indicándole indirectamente a un resolver validador si un dominio existe o no. Veamos un ejemplo:

Ejemplo 6-1: RR NSEC

```
uy. 86400 IN NSEC next.uy SOA A NS RRSIG NSEC DNSKEY
```

Este registro NSEC le pertenece al dominio uy –el ápex de la zona–. La etiqueta que sigue a NSEC –next.uy–, indica el siguiente dominio que existe en la zona luego de uy, en el orden canónico que es alfanumérico. Los siguientes nombres de recursos que aparecen indican los recursos que están disponibles al momento de firmar, para el nombre de dominio de la primera etiqueta. De esta manera un servidor autoritativo puede demostrar de forma segura si un nombre de dominio no existe –NXDOMAIN–, o también indicar qué sí existe, pero que no tiene el RR solicitado –NODATA–. Se debe notar que para que esto funcione, la zona debe estar ordenada de forma canónica, de manera que los intervalos vacíos puedan ser unívocamente identificados. Todas las herramientas que firman las zonas, como dnssec-signzone de BIND ordenan canónicamente el archivo automáticamente, de modo que esto no representa una dificultad adicional. El último NSEC de la zona apuntará al primer NSEC, que corresponde al del ápex de la zona, cerrando así la cadena de nombres.

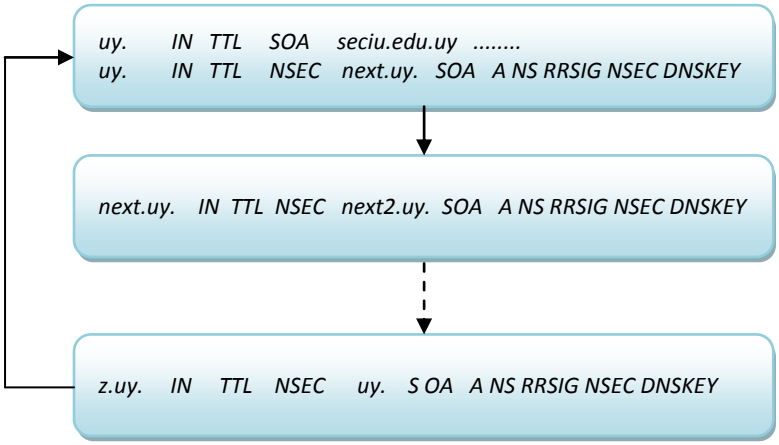


Figura 6.1: Cadena NSEC

Cuando un resolver security-aware realice una consulta a un servidor autoritativo de la zona uy por un nombre de dominio no existente, por ejemplo ab.uy siguiendo el esquema de la Figura 6.1, recibirá la siguiente respuesta:

Ejemplo 6-2: Respuesta NXDOMAIN

```
;; status: NXDOMAIN, id: 53215

;; AUTHORITY SECTION:
uy.          SOA          ...
uy.          RRSIG        SOA  ...
uy.          NSEC         next.uy  SOA A NS RRSIG NSEC DNSKEY
uy.          RRSIG        NSEC   ...
```

En este caso, seguiremos recibiendo el estado NXDOMAIN, con el objetivo de mantener compatibilidad hacia atrás, pero debemos tener en cuenta que en DNSSec no confiamos en los encabezados. También recibimos el registro SOA con el mismo principio. Para que el resolver pueda efectivamente concluir que dicho dominio no existe, debe verificar que el RR NSEC que recibe *cubre* el dominio consultado, en este caso ab.uy y validar su firma digital, el RRSIG del NSEC. Se dirá que un registro NSEC *cubre* a un nombre de dominio si este se encuentra ordenado canónicamente entre el nombre del NSEC y el nombre indicado en este registro como el próximo nombre valido.

Analicemos ahora cómo es una respuesta NODATA con NSEC. Siguiendo con el mismo ejemplo, supongamos que se realiza una consulta por el RR TXT de next.uy. Se obtendrá por lo tanto la siguiente respuesta:

Ejemplo 6-3: Respuesta NODATA

```
;; status: NOERROR, id: 24128

;; AUTHORITY SECTION:
uy.          SOA          ...
uy.          RRSIG        SOA  ...
next.uy.     NSEC         next.uy  SOA A NS RRSIG NSEC DNSKEY
next.uy.     RRSIG        NSEC   ...
```

A diferencia del caso anterior, no tenemos un código para indicar en el encabezado el estado NODATA –aunque igual no significaría mucho en DNSSec como ya sabemos–. El resolver debe inferir por lo tanto que está en presencia de un NODATA observando lo siguiente: el dominio next.uy sí existe, ya que tiene asociado un registro NSEC y que el registro TXT no existe ya que no está listado en dicho registro NSEC. Si la firma digital es validada correctamente, ya tenemos casi completada nuestra autenticidad de no existencia con NSEC.

6.2.1 - Wildcards

Lo único que falta para completar el análisis de NSEC, es el problema de autenticar una respuesta generada por una wildcard. Los wildcard son parte de DNS desde sus inicios, pero varias de sus definiciones originales han sido cambiadas y mejoradas con el tiempo, resultando finalmente en la RFC 4592 [7]. Veamos algunas de ellas que utilizaremos más adelante junto con otras definiciones propias de DNS –RFC 1034 [3]− que ayudan a definir estas nuevas funcionalidades.

QNAME: Este término especifica el nombre de dominio objetivo de una consulta.

QTYPE: Es el tipo de recurso solicitado en la consulta: A, MX, etc.

Etiqueta asterisco: Se dice que un nombre de dominio tiene una etiqueta asterisco si este empieza con un "*". La primera etiqueta del nombre debe ser solo el asterisco. Ejemplo: *.edu.uy.

Nombre de dominio Wildcard: Esto es un nombre cuya primera etiqueta es una etiqueta asterisco. Cuando decimos que existe una wildcard en la zona, nos referiremos a un nombre de dominio wildcard. Según se especifica en la RFC 4592 [7], no hay ninguna diferencia entre un nombre de dominio normal y una wildcard, ya que están codificadas como registros de recursos, por lo que deben ser tratadas como tal a efectos de transferencias de zona por ejemplo.

Closest encloser: Dada un nombre de dominio, su closest encloser es el nodo del árbol de dominio DNS con el que tiene más etiquetas en común, contando desde la raíz hacia abajo. Por definición, el closest encloser es un nombre válido y existente en la zona. Ejemplo: el closest encloser de abc.def.edu.uy es edu.uy –suponiendo que def.edu.uy y abc.def.edu.uy. no existen−.

Next closer name: Es el closest encloser más una etiqueta agregada a la izquierda. Si edu.uy es el closest encloser de la consulta abc.def.edu.uy, el next closer name es def.edu.uy. Esta definición es introducida en la RFC 5155 [20], que trata específicamente NSEC3.

6.2.1.1 - Sintetizar una respuesta wildcard

Las wildcard le permiten a DNS definir todos los nombres para cierto tipo de una sola vez. Un uso muy común de esta configuración es usarlo para resolver todos los host en una zona. En el Ejemplo 6-4, el servidor resolverá todos los subdominios de universidad.edu.uy menos el de ftp.universidad.edu.uy.

Ejemplo 6-4: Utilización de wildcard

```
*.universidad.edu.uy      IN     TTL    A      10.0.0.1
ftp.universidad.edu.uy    IN     TTL    A      10.0.0.2
```

Si un resolver pregunta por www.universidad.edu.uy, el servidor de nombres autoritativo de edu.uy −suponiendo que el dominio universidad.edu.uy no ha sido delegado− sintetizará una respuesta, o se dice también que expandirá la wildcard, para devolver la siguiente respuesta:

Ejemplo 6-5: Respuesta

```
;; status: NOERROR, id: 532

;; ANSWER SECTION:
www.universidad.edu.uy    A      10.0.0.1
```

Anteriormente a DNSSec un resolver no podía, ni necesitaba, detectar que esta respuesta provenía de una wildcard, por lo que para validar una respuesta de este tipo con DNSSec debemos indicar de alguna manera que la respuesta se generó expandiendo una wildcard, y que además no existe www.universidad.edu.uy definido explícitamente con su propio registro A. Veamos ahora como sería la respuesta anterior suponiendo que el servidor autoritativo estuviera firmado:

Ejemplo 6-6: Respuesta firmado

```
;; status: NOERROR, id: 26051

;; ANSWER SECTION:
www.universidad.edu.uy    A      10.0.0.1
www.universidad.edu.uy    RRSIG  A 5 3 ...

;; AUTHORITY SECTION:
yyy.universidad.edu.uy    NSEC zzz.universidad.edu.uy A RRSIG NSEC
yyy.universidad.edu.uy    RRSIG NSEC ...
```

En primer lugar se debe notar que el RRSIG de www.universidad.edu.uy no existe, si no que el que se devuelve es el RRSIG de la wildcard –recordemos que DNSSec no está pensado para crear firmas a demanda−. El resolver entonces reconoce que la respuesta es producto de una expansión de wildcard, observando la discrepancia en el contador de etiquetas del RRSIG, que en este caso indica 3 cuando debería indicar 4. Este contador indica el número de etiquetas del nombre de dominio al que pertenece el RRSIG, excluyendo la etiqueta wildcard y a la raíz. Recordemos que la firma digital incluye los demás parámetros del RRSIG, incluyendo este contador de etiquetas. Sabiendo ahora que la respuesta provino de una wildcard, debemos ver ahora como se autentica.

6.2.1.2 - Validar una wildcard con NSEC

Para que el resolver pueda validar que la respuesta generada por la wildcard es auténtica, debe estar seguro que no existe el dominio consultado, ni una wildcard más especifica. Aquí entra a jugar el NSEC. En el ejemplo anterior observamos que el registro NSEC cubre a www.universidad.edu.uy, por lo que sabemos que la wildcard está correctamente usada dado que no existe efectivamente el QNAME.

Supongamos que ahora un resolver pregunta por el registro A de ftp.universidad.edu.uy y dado que existe uno para dicho dominio, no es necesario sintetizar una respuesta. Supongamos también que no incluimos un NSEC en la respuesta, porque no lo consideramos necesario. Dada esta situación, un atacante podría modificar la respuesta de la siguiente manera:

Ejemplo 6-7: Respuesta corrupta

```
;; status: NOERROR, id: 14798

;; ANSWER SECTION:
ftp.universidad.edu.uy    A     10.0.0.1
(Notar que la IP no es la correcta según se definió en el Ejemplo 6-6)
ftp.universidad.edu.uy    RRSIG  A 5 3 ...
```

El atacante intenta hacer creer al resolver que la respuesta se generó con una wildcard, oscureciendo el hecho que en realidad existe un A RR para ftp.universidad.edu.uy. Y el ataque hubiera sido exitoso de no ser que el estándar DNSSec exige devolver un NSEC –RFC 4035 [14]–, cuando una respuesta se sintetiza a partir de una wildcard, para probar que no existe un nombre más específico en que la respuesta se pudiese haber basado. De esta manera, si un resolver security-aware recibe una expansión de wildcard pero sin un NSEC, no validará los datos.

6.2.1.3 - Validar la no existencia de una wildcard

Ya hemos visto que con NSEC podemos validar respuestas NXDOMAIN, devolviendo un registro que cubra el dominio solicitado. Sin embargo, también se debe demostrar que no existe una wildcard capaz de sintetizar la respuesta deseada, por lo tanto se requiere otro NSEC más para asegurar esto. Veamos ahora un ejemplo real de la zona raíz. Realicemos una consulta por el dominio hola.:

Ejemplo 6-8: Consulta real

```
dig hola. +dnssec
```

Indicando la flag +dnssec, le pedimos al servidor que devuelva todos los registros necesarios para validar la respuesta. Obtenemos el siguiente fragmento:

Ejemplo 6-9: Respuesta

```
;; status: NXDOMAIN, id: 14523

;; QUESTION SECTION:
;hola.                    IN      A

;; ANSWER SECTION:
.            SOA          a.root-servers.net
.            RRSIG        SOA 8 0  ...
.            NSEC         ac. NS SOA RRSIG NSEC DNSKEY
.            RRSIG        NSEC 8 0 ...
hn.          NSEC         hr. NS DS RRSIG NSEC
hn.          RRSIG        NSEC 8 1 ...
```

Aquí tenemos que la respuesta contiene dos registros NSEC, el primero es para indicar que no existe una wildcard que pudiese haber sintetizado la respuesta. Tener

en cuenta que en el orden canónico de nombres DNS, "*." esta está entre "." y "ac.". El segundo NSEC es el que ya conocemos e indica que efectivamente el dominio hola no existe en la raíz.

Con esto tenemos que para autenticar la no existencia de un dominio, tenemos que autenticar que no pertenece a la zona y que no puede ser generado por una wildcard, lo que requiere como máximo dos NSEC. Se puede dar el caso particular en que el dominio consultado y la wildcard sean cubiertos por el mismo NSEC, en cuyo caso solo un registro NSEC es necesario.

6.2.2 - Problemas de NSEC

Si bien hemos logrado tener autenticidad de no existencia de dominio –NXDOMAIN y NODATA−, esta extensión de seguridad introdujo un efecto secundario denominado enumeración de zona. La enumeración permite a un usuario externo a la administración de la zona, obtener todos los datos de un archivo de zona cuya transferencia –AXFR, ver Anexo F− está limitada, degradando así el nivel de confidencialidad que algunos dueños de dominio desean. Debemos recordar en este punto que los datos de DNS son siempre públicos, pero la enumeración permite de forma muy sencilla la obtención de todos los datos de la zona, lo que significa una nueva brecha de seguridad. Anteriormente a la introducción de NSEC, la enumeración no era sencilla y requería fuerza bruta para identificar los dominios de la zona, sin estar seguros nunca de tenerlos realmente a todos. Con este nuevo registro, simplemente se tiene que buscar registros que no existan y rescatar del recurso NSEC el próximo nombre de dominio de la zona.

Otro inconveniente que se observó con NSEC es que cambiar unos pocos nombres de dominio podría desencadenar que gran parte de la cadena NSEC se tuviera que construir devuelta, y por lo tanto también sus firmas. La carga de CPU requerida para hacer esto puede resultar excesiva, sobre todo en zonas con alto contenido de registros NS, delegaciones, como las TLD y ccTLD. Generalmente, las zonas hijas de estas no estarán firmadas en las primeras etapas de DNSSec, por lo que la reconstrucción de la cadena NSEC resultaría innecesaria ya que no aporta ningún tipo de seguridad a esas zonas hijas. El método de autenticidad de no existencia NSEC3 incorpora lo que se denomina Opt−Out, un mecanismo que le permite al operador de zona evitar reconstruir toda la cadena salteándose de la misma los registros NSEC3 de las delegaciones inseguras. Detallaremos más adelante los usos de Opt−Out.

Para que DNSSec fuera aceptada y desplegada con éxito, debía proveer algún mecanismo adicional que evitara la enumeración trivial de zona y permitiera el incremento gradual de delegaciones seguras a la cadena de no existencia. Surge entonces un nuevo registro, el NSEC3, el cual no sustituye a NSEC, sino que ambos mecanismos de PNE −Prueba de No Existencia− son aceptados por DNSSec y el administrador de la zona puede optar entre ambas posibilidades. Por ejemplo, la zona raíz está firmada utilizando NSEC, mientras que la com utiliza NSEC3.

6.3 - NSEC3

Si bien hubo otras alternativas previas a NSEC3, como los registros NO, DNSNR y por supuesto NSEC2, finalmente la nueva especificación de DNSSec se quedó con NSEC3, la cual surge de la experiencia y resultados de estos registros anteriores que aportaron sus mejores aspectos al éxito de NSEC3, entre estos los hashes y Opt−out.

La RFC 5155 [20] es el documento principal en donde se especifican las funciones, requerimientos y consideraciones que se deben adoptar con NSEC3. Rescataremos aquí los aspectos importantes que aporta esta RFC y dejaremos los detalles, como la codificación de los datos en el cable, como lectura complementaria en el Anexo C.

6.3.1 - Nuevo espacio de nombres

Para solucionar el problema de la enumeración de la zona, los registros NSEC3 utilizan hash de nombres de dominios para todos los nombres, tanto para el dueño del recurso como para indicar el próximo dominio. Veamos un ejemplo de este recurso de la zona com.:

Ejemplo 6-10: NSEC3

```
3RL0HJVPUVPJAI1.com. TTL IN NSEC3 1 1 0 SALT 3RRLL7B5O5S514OE NS DS RRSIG
```

Todos hash presentes en los NSEC3 responden a una fórmula iterativa, la cual utiliza 3 parámetros: el nombre de dominio original, número de iteraciones y la salt.

```
IH(salt, x, 0) = H(x || salt)
IH(salt, x, k) = H(IH(salt, x, k-1) || salt), si k > 0

IH(salt, nombre de dominio original, iteraciones)
```

En el contexto de NSEC3, se denomina nombre de dominio original al nombre totalmente calificado −FQDN− y en minúsculas antes de realizar la operación del hash. En caso de que el nombre de dominio sea una sintetización de una wildcard, su nombre original es el nombre sin expandir e incluyendo la etiqueta wildcard. El número de iteraciones es la cantidad de veces que se itera el algoritmo, los hashes sucesivos utilizan el resultado del paso anterior. Si el número de iteraciones es 0 –como en el ejemplo− se realiza una sola vez la operación de hash, el paso cero del algoritmo. El parámetro salt –sal− es un string que se agrega al nombre de dominio antes de la realización del hash. El nombre de este parámetro seguramente venga del hecho de que su objetivo es “aderezar” el nombre de dominio. En el ejemplo propuesto, el registro NSEC3 indica que la salt es el string “SALT”, sin embargo, la zona com no hace uso de este parámetro y esto se indica con un guion “–“ en el campo de la salt. Un poco más adelante veremos algunos ejemplos de parámetros salt elegidos por administradores de zonas, así como el número de iteraciones.

La función hash H(.) utilizada en el algoritmo es la SHA-1 y hasta la fecha es la única que tiene DNSSec estandarizada para la realización de los hash de NSEC3. Dado que se puede considerar que NSEC3 es relativamente nuevo en relación a DNSSec, se debió considerar que podían existir resolvers validadores que no reconocieran NSEC3, por lo que fallarían en su intento de reconocer estos nuevos registros. Para evitar esto se introdujeron dos nuevos identificadores de algoritmos para las DNSKEY, con el propósito de simplemente señalizar que la zona está firmada utilizando NSEC3, sección 2 RFC 5155 [20]. Una tabla presentando estos identificadores se puede encontrar en la sección 7.2.2 - . Un resolver que no reconoce NSEC3 dará entonces por inseguros los datos de la zona.

Volviendo con nuestro ejemplo. El nombre dueño de un registro NSEC3 será entonces el hash del nombre original, seguido del ápex de la zona, que en el ejemplo sería .com. Esto no tiene otro propósito más que darle mejor legibilidad al registro. El segundo hash, el que indica el próximo hash de nombre, no contiene la etiqueta del ápex de la zona, pero de ninguna manera se diferencia en algo su construcción. Una

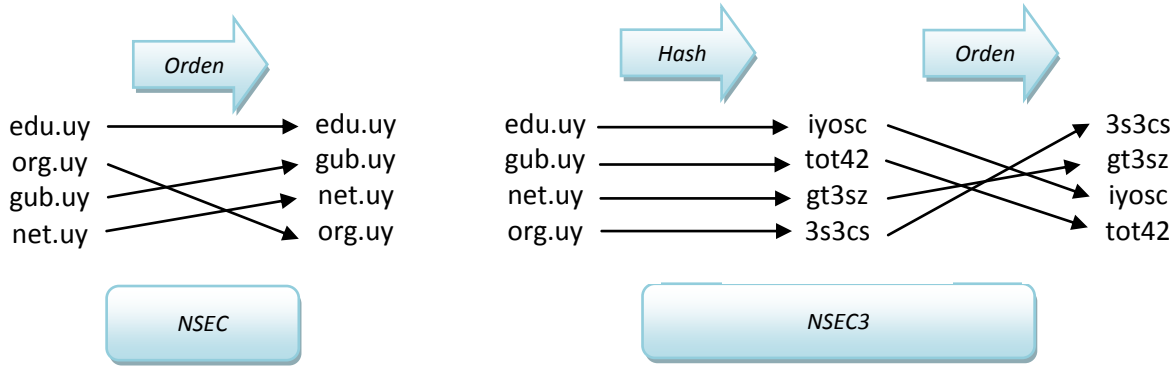


Figura 6.2: Cadena de NSEC y NSEC3

pregunta válida que corresponde hacerse en este momento, es como se ordenan estos nuevos registros. Como vimos en NSEC, es necesario ordenar los dominios de forma canónica de modo de tener identificados todos los vacíos de nombres correctamente. Sin embargo, dado que NSEC3 usa hash de nombres, ordenar los registros canónicamente resultará en principio algo un poco desordenado.

En la Figura 6.2, tenemos el espacio de nombres ordenados según el orden canónico para NSEC y luego según el orden canónico después de realizado el hash. En este último caso, NSEC3, diremos que se generó un nuevo espacio de nombres, el espacio de nombres hash. Los registros NSEC3 encadenan estos últimos hash luego de ordenarlos canónicamente y el último registro apuntará al primero, pero a diferencia de NSEC esta vez no será el ápex de la zona –con alta probabilidad al menos–.

Al igual que con NSEC, debemos seguir siendo capaces de brindar prueba de no existencia NXDOMAIN y NODATA, y además cubrir la posibilidad de wildcards. Pero a diferencia de NSEC, se precisarán un máximo de tres registros NSEC3 para hacer lo mismo. La razón de este tercer registro, se debe a que los hash de nombre de dominios generan un espacio de nombres plano. Esto significa que perdemos la profundidad de la zona en el árbol de nombres DNS y con ella, información acerca del closest encloser fundamental para determinar la no existencia de una wildcard.

6.3.2 - Opt−Out

Como ya hemos adelantado NSEC3 incorpora una nueva facilidad, que le permite al operador de zona saltearse de la cadena de NSEC3 las delegaciones inseguras[3], permitiendo que estas puedan ser removidas o agregadas sin necesidad de refirmar la zona ni modificar la cadena NSEC3. Además, en zonas centro-delegadoras, como las ccTLD las cuales son mayormente dominadas por registros NS, Opt−out puede resultar en un ahorro de memoria y CPU ya que no se crearían tantos NSEC3 ni se realizarían sus hashes ni sus firmas criptográficas. Un registro NSEC3 Opt−out se identifica por tener la flag opt-out en 1, indicado que este registro cubre una o más delegaciones inseguras. Una zona puede contener una mezcla de NSEC3 "comunes" o NSEC3 Opt−Out. Los primeros registros, no pueden cubrir ningún nombre de dominio de delegación insegura. Los Opt−Out, al contrario, solo pueden cubrir nombres de dominio de dichas delegaciones inseguras.

6.3.3 - Prueba del Closest Encloser

Muchas de las respuestas que involucran registros NSEC3 utilizarán el concepto de Prueba del Closest Encloser. Como ya vimos, el espacio de nombres hash carece de "profundidad", por lo que un resolver que intente validar una no existencia, debe saber cuál es su closest encloser para poder verificar que no existe una wildcard en ese nivel del árbol de nombres originales –sin hash−.

Esta prueba consiste en hasta dos registros NSEC3:

1. Un NSEC3 que corresponda al closest encloser.
2. Un NSEC3 que cubra el next closer. Este registro puede llegar a coincidir con el primero en el caso particular de que el primero cubra su next closer.

Estos dos registros son denominados la Prueba del Closest Encloser y prueban que no existe un nombre de dominio más específico al closest encloser y que dicho dominio sí existe.

Para ilustrar esta parte, nos definiremos una zona de autoridad con la que trabajaremos en esta sección.

[3] Una delegación insegura se caracteriza por tener un registro NS, pero no uno DS.

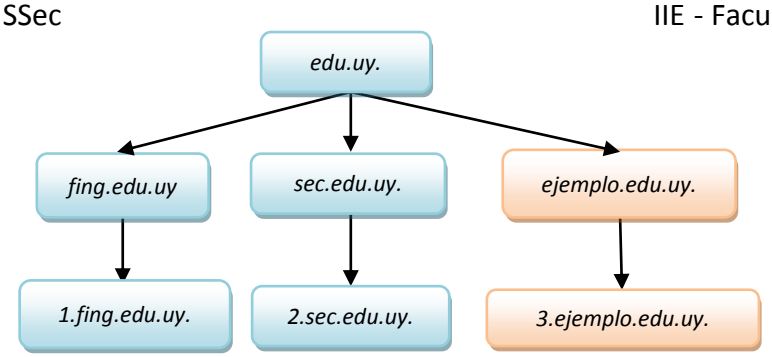


Figura 6.3: Zona de autoridad

Los dominios en azul representan dominios existentes en nuestra zona, los rosados serán los dominios que no existen. Según vimos anteriormente, debemos formar el nuevo espacio de nombres hash correspondiente a esta nueva zona, para luego ordenarlos canónicamente. La siguiente tabla muestra los hash de los nombres de la Figura 6.3 y sus órdenes respectivos en la cadena NSEC3. Las últimas dos entradas de la tabla muestran los hash de los nombres que necesitaremos a continuación, así como el intervalo –Registro NSEC3– que cubre el nombre hash correspondiente.

Tabla 6-1: Espacio de nombres hash de la zona.

Nombre	Hash	Orden en la cadena NSEC3
1.fing.edu.uy.	LD03LS3JSFK2	5
2.sec.edu.uy.	11AKF9FL234S	1
edu.uy.	AFLD93L20DLD	3
fing.edu.uy.	D5LGS0L2L30D	4
sec.edu.uy.	164DK03LSD0F	2
ejemplo.edu.uy	Q4DKG2L30DL2	Fuera de zona: 5–1
*.edu.uy.	8SDLFE2333LS	Fuera de zona: 2–3

Las consultas realizadas por dominios inexistentes en la zona de la Figura 6.3 devolverán registros NSEC3. Supongamos que un resolver validador consulta por el dominio 3.ejemplo.edu.uy., y recibe los siguientes registros:

Ejemplo 6-11: Respuesta a 3.ejemplo.edu.uy

```
1)  AFLD93L20DLD.edu.uy.   NSEC3  1  0  2 SALT   D5LGS0L2L30D SOA NS ...
2)  LD03LS3JSFK2.edu.uy.   NESC3  1  0  2 SALT   11AKF9FL234S A ...
3)  164DK03LSD0F.edu.uy.   NSEC3  1  0  2 SALT   AFLD93L20DLD A ...
```

La primera tarea del resolver es encontrar el closest encloser, para ello debe tomar sucesivos fragmentos del QNAME, realizarle el hash –incluyendo el salt y las iteraciones–, hasta encontrar un hash que se corresponda con el dueño de alguno de estos NSEC3. El primer paso para esto es realizar el hash al QNAME entero, este es 3.ejemplo.edu.uy y verificar si corresponde a algunos de los NSEC3 anteriores. Dado que no verifica, quitamos la última etiqueta y probamos devuelta, en este caso ejemplo.edu.uy tampoco es el closest encloser. Probamos nuevamente removiendo la etiqueta ejemplo y verificamos que esta vez sí tenemos una correspondencia. El primer

registro NSEC3 corresponde con el hash de edu.uy., por lo que ahora el resolver tiene un punto de partida.

La segunda tarea del resolver es comprobar la no existencia del next closest encloser, como lo definimos al principio de este capítulo. Para esto el resolver parte del closest encloser –edu.uy– y le agrega una etiqueta más, resultando en ejemplo.edu.uy. El hash de este nombre de dominio debe estar cubierto por alguno de los otros NSEC3 y como vemos en la tabla de hashes esto se verifica con el segundo registro NSEC3 del ejemplo. Con esto el resolver sabe que existe edu.uy. y que no existe ejemplo.edu.uy. A estos dos primeros registros se les denomina la Prueba del closest encloser.

Un escenario alternativo al que plantemos anteriormente, es que usando Opt−Out no se pueda probar la existencia del closest encloser, ya que es parte de una delegación insegura cubierta por un NSEC3 Opt−Out. En este caso, en vez de probar la existencia del closest encloser, se prueba lo que se denomina como el closest probable encloser, el cual es el ancestro más específico que se puede probar que existe.

6.3.4 - NXDOMAIN con NSEC3

Para probar la no existencia de un dominio solicitado, el servidor autoritativo debe proveer la prueba del closest encloser –hasta dos NSEC3–, más un NSEC3 cubriendo la no existencia de una wildcard en el closest encloser. Esto se realiza agregándole al closest encloser la etiqueta asterisco –"*.closest_encloser."–, realizando el hash a dicho nombre resultante y verificando que existe un registro NSEC3 cubriéndolo. Lo que suma un máximo de tres registros NSEC3 necesarios para probar la no existencia NXDOMAIN. Desde el punto de vista de un resolver validador, este debe realizar las operaciones necesarias para verificar todos los registros que recibe. Esto incluye verificar la prueba del closest encloser, la cual consiste como ya vimos en la realización de varios –dependiendo del número de etiquetas del QNAME– hashes en la búsqueda del closest encloser, verificar la prueba del next closest encloser y por último generar el hash para verificar el wildcard. Todo esto significa una sobrecarga en ancho de banda y más importante en CPU, del servidor autoritativo y más aún del resolver validador. El impacto que tiene esto sobre la performance de un resolver lo estudiaremos más adelante.

6.3.5 - Prueba NODATA

El principio para esta prueba es igual que en NSEC. Para indicar que un recurso no pertenece a un nombre de dominio, se debe enviar un NSEC3 cuyo dueño sea el hash del QNAME y que no contenga el RR requerido en el campo de registros. El resolver validará esta respuesta verificando la firma digital del NSEC3, habiendo chequeado anteriormente que el registro corresponde efectivamente con el dominio en cuestión.

En el caso particular de que se solicite el recurso DS se debe tener cuidado si se está utilizando Opt-Out. Si es así, no debería existir un NSEC3 que corresponda al QNAME, por lo que el servidor deberá proveer la prueba del closest probable encloser, y un NSEC3 que cubra el next closer y tenga el bit opt-out, indicando que la delegación no es segura y que por lo tanto no existe este recurso.

6.3.5.1 - Prueba NODATA con Wildcards

Puede ser posible que exista un wildcard capaz de generar el nombre de dominio solicitado, pero sin embargo que el recurso pedido –QTYPE– no esté disponible. Para demostrar esto, el servidor deberá proveer una prueba del closest encloser del QNAME y un NSEC3 que corresponda a la wildcard generadora. De esta manera se demuestra que el QNAME no existe pero que sí existe una wildcard generadora. Un resolver validador será capaz de autenticar la respuesta verificando los hashes de los NSEC3 y validando las firmas digitales de los mismos.

6.3.6 - Respuestas positivas con Wildcads

Siendo más optimistas, veamos ahora que pasa cuando el QNAME y QTYPE sí existen, pero son resultado de una sintetización de wildcard. Al igual que con NSEC, un resolver validador identifica que la respuesta provino de una wildcard observando la diferencia entre el contador de etiquetas en el RRSIG y el QNAME, por lo que es necesario probar si la wildcard fue correctamente utilizada.

El servidor de nombre autoritativo debe probar que el QNAME no existe y que su closest encloser y el ancestro inmediato de la wildcard son iguales, probando así que la wildcard fue correctamente utilizada. La prueba de esto consiste en un solo NSEC3 que cubra el next closer del ancestro inmediato de la wildcard.

6.3.7 - NSEC3 en los Caches

Con NSEC un cache es capaz de identificar el registro NSEC apropiado de su cache para enviar en respuesta, ya que el nombre de dominio del mismo no está oscurecido y no es necesaria ninguna operación criptográfica para identificarlos. Sin embargo, con NSEC3 esto es diferente. Un cache es incapaz de realizar este tipo de operaciones por lo que no será capaz de identificar los registros necesarios de su cache. En principio, de no implementarse nuevos mecanismos para la realización de los caches, esto significaría un incremento en el tráfico y carga de los servidores autoritativos.

6.4 - NSEC3PARAM

El NSEC3PARAM es un recurso nuevo, definido en la RFC 5155 [20], el cual se utiliza para almacenar los parámetros propios de NSEC3: algoritmo, iteraciones y salt. Este recurso siempre pertenece al ápex de la zona y sirve de referencia para los servidores autoritativos esclavos acerca de cuáles son los parámetros a utilizar al momento de generar hashes parar elegir correctamente los NSEC3 que se deben devolver. Aunque este recurso pertenece a la zona y se puede obtener fácilmente, no debe ser utilizado por los resolver validadores para obtener los parámetros de NSEC3.

Puede existir más de un RR NSEC3PARAM en la zona, pero por cada uno debe existir también una cadena completa de NSEC3 utilizando los parámetros declarados en cada NSEC3PARAM. El servidor autoritativo, maestro o esclavo, puede elegir entre cualquier juego de parámetros para realizar y buscar los NSEC3 necesarios en la respuesta. Veamos un ejemplo de este nuevo registro de recurso.

Ejemplo 6-12: NSEC3PARAM

```
example.com.   NSEC3PARAM  1  0  12  aabbccdd
```

Se desprende de este ejemplo que el ápex de la zona es example.com, que el algoritmo utilizado es el RSASHA1 –el único disponible para NSEC3– el número de iteraciones a realizar son 12 y que el salt es aabbccdd.

6.4.1 - Iteraciones

El propósito de las iteraciones no es otro que fijar el costo computacional de generar los hashes, determinando que un ataque diccionario consuma tiempo y recursos adicionales. Un ataque diccionario consiste en que el atacante primero adquiera todos los registros NSEC3 de la zona –lo cual no presenta ninguna dificultad–, luego calcule utilizando los parámetros del NSEC3 todos los hashes de los posibles nombres de dominio de la zona –de ahí el nombre diccionario–, y por último compararlos con los hashes dueños de los NSEC3, enumerando así la zona. Evidentemente este método es mucho más costoso que con NSEC y este costo lo impone el número de iteraciones que determina el tiempo y recursos que el atacante debe disponer para generar el diccionario de hashes.

Dado que el número de iteraciones afecta tanto al servidor autoritativo como a los resolvers validadores, se debe imponer un límite superior de iteraciones. Se determinó –RFC 5155 sección 10.3– que debía seguirse el siguiente delineamiento al momento de elegir el número de iteraciones.

Tabla 6-2: Iteraciones máximas

Tamaño de la Clave (bits)	Iteraciones
1024	150
2048	500
4096	2500

En función del tamaño de la clave KSK más chica de la zona, el máximo número de iteraciones queda determinado por la Tabla 6-2. El tamaño de clave se redondea para arriba, claves entre 1024 y 2048 bits de longitud pueden utilizar un máximo de 500 iteraciones.

La tabla está basada en una aproximación de la relación del costo entre calcular un SHA−1 y verificar una firma RSA. Verificar una firma digital de una clave de 1024 bits equivale computacionalmente a realizar 150 hashes SHA−1.

6.4.2 - Salt

La función de la salt es también la de prevenir ataques diccionario, pero a diferencia de las iteraciones, esta no afecta el costo de generar un diccionario, si no que impone que el diccionario tenga que ser generado nuevamente cada vez de se cambia la salt. Por lo que cambiar la salt de manera regular obliga al atacante a comenzar nuevamente, dificultando aún más la enumeración de zona. El uso de la salt no es obligatorio, como hemos visto la zona com que está firmada con NSEC3, no la utiliza, ni tampoco las iteraciones –una iteración siempre se hace–. Por lo que el periodo de vida de un salt no está especificado en DNSSec y queda a decisión del administrador de la zona ya que es un trade-off entre cambiarlo muy seguido incurriendo en costos, o cambiarlo cada más tiempo arriesgando la enumeración de zona.

7. Claves DNSSec

7.1 - Introducción a las claves DNSSec

Hasta ahora vimos que DNSSec introduce nuevos registros al protocolo DNS, los cuales son procesados por los resolvers sin necesidad de intervención humana. Sin embargo, existe un aspecto del protocolo que no escapa al contacto humano, las claves. Si bien en teoría DNSSec soluciona los problemas de autenticidad e integridad entre servidor-cliente, debe también proveer mecanismos –recomendaciones− para asegurar la parte más vulnerable del sistema, el usuario u operador.

Como ya vimos, se utilizan claves públicas y privadas, las cuales deben ser generadas, almacenadas y cambiadas bajo la supervisión del responsable de la zona. Si bien estos procedimientos deben ser automatizados pueden existir razones para realizarlas manualmente. Estos procedimientos deben estar documentados y registrados asegurando la mayor transparencia y seguridad de los mismos. Y como bien lo indica el nombre del método para autentificar claves, Cadena de Confianza, se debe tener en cuenta dos cosas: DNSSec se basa en la confianza que depositamos en ciertas instituciones para que administren las claves que utilizaremos, y que sin importar sus niveles jerárquicos −en el árbol de dominios−, cada una es como un eslabón y es tan importante para sus subdominios como el mismo *root*.

A continuación definiremos estos métodos, y recomendaciones, que el protocolo establece para gestionar y administrar todos los aspectos que involucren decisiones y contacto de personal.

7.2 - Generación de claves

7.2.1 - Actores

En primer lugar trataremos sobre los responsables de generar las claves, quienes son, sus funciones y que rol cumplen en DNSSec.

Se denomina a la institución responsable de generar las claves como el *Key Operator*, el protocolo propone un operador distinto para la KSK y la ZSK de cada zona, sin embargo, una sola institución para ambas es también admitida. El operador de la KSK de la zona es responsable de las siguientes operaciones:

- Generar y proteger la componente privada de la KSK.
- Importar de forma segura la parte publica de las ZSK de su operador.
- Autentificar y validar las claves anteriores
- Firmar de forma segura los RRset que contengan estas ZSK.
- Transmitir de forma segura la firma digital del RRset de la ZSK a su operador.
- Exportar de forma segura la parte pública de la KSK.
- Realizar un roll-over de emergencia en un tiempo razonable si la componente privada de la KSK se pierde o se compromete.

En caso de que haya operadores distintos para la KSK y la ZSK, el operador de esta última deberá responsabilizarse por las siguientes tareas:

- Generar y proteger la componente privada de la ZSK.
- Exportar y transmitir de forma segura la parte publica de la ZSK al operador de la KSK.
- Importar de forma segura la firma digital del RRset de la ZSK del operador de la KSK.
- Firmar los RRset de la zona.
- Realizar un roll-over de emergencia en un tiempo razonable si la componente privada de la ZSK se pierde o se compromete.

Por otro lado, si se desea incluir la zona en una cadena de confianza ya existente, el administrador de la zona es responsable también de mantener en la zona padre, el SEP −Security Entry Point− o el RR DS de la KSK.

En la raíz, tenemos que el operador de la KSK es la ICANN, mientras que el operador de la ZSK es VeriSign. La distinción entre ambos operadores está asociada a la distinción que se hace entre la KSK y la ZSK. Si bien ambas funcionan exactamente igual, como ya se discutió, la diferencia está en sus usos. Como con los operadores, la distinción ente ambos operadores es de carácter más político, y el administrador de cada zona elige delegar la administración de las claves de la forma que más encuentre adecuado.

Es importante destacar nuevamente, que el rol que estas instituciones juegan en DNSSec está basado en la confianza que los usuarios finales de Internet depositan en las mismas. El administrador de la zona raíz es la Administración Nacional de Telecomunicaciones e Información −NTIA por sus siglas en Inglés−, agencia bajo la

supervisión del Departamento de Comercio de Estados Unidos, la cual a su vez delega la operación de la raíz a ICANN. Siguiendo esta línea de razonamiento, reconocemos que Internet en general, y más aún DNSSec, se basa en que confiamos en que estas instituciones realizarán sus responsabilidades de forma correcta.

7.2.2 - Algoritmos

Tanto las claves como los hash utilizados para la firma digital están estandarizados en DNSSec, ya que cada RRSIG debe indicar que algoritmo se utilizó en la firma. La siguiente tabla muestra los posibles algoritmos aceptados por DNSSec.

Tabla 7-1: Algoritmos

Valores	Algoritmo
0	Reservado
1	RSA-MD5
2	Diffie-Hellman (RFC 2539)
3	DSA/SHA-1—Opcional (RFC 3755, 2536)
4	Elliptic curve—Todavía no estandarizado
5	RSA/SHA-1—Obligatorio (RFC 3755, 3110)
6	DSA-NSEC3-SHA1 (RFC 5155)
7	RSASHA1-NSEC3-SHA1 (RFC 5155)
8	RSA/SHA-256 (RFC 5702)
9	Sin Asignar
10	RSA/SHA512 (RFC 5702)
11	Sin Asignar
12	GOST R 34.10-2001 (RFC 5933)
13-122	Sin Asignar
123-251	Reservado
252-254	Sin Asignar
255	Reservado

El valor del algoritmo utilizado es indicado en el campo RDATA de cada RRSIG. El más común y frecuentemente utilizado es el 5, sin embargo, se recomienda utilizar el 8 o el 10, debido a que se detectaron en este primero fallas criptográficas.

En general la vida útil de estos algoritmos está determinada por el avance del criptoanálisis y el aumento del poder de cálculo de las computadoras, un algoritmo puede haber sido perfectamente seguro algunos años atrás, pero hoy son considerados hasta peligrosos de usar. Un ejemplo es el algoritmo de hash con valor 1, MD5. En 1991, releva a su antecesor MD4 para proveer integridad de datos, sin embargo, en 1996 se anuncian colisiones en la función compresión de este hash. En el 2004 finalmente se publican colisiones de hash MD5 en menos de una hora, lo que provocó que nuevos métodos se comenzaran a utilizar [25].

El SHA −Secure Hash Algorithm−, fue publicado en 1993 por la NSA −Agencia Nacional de Seguridad, EEUU−. En 1998 se encuentra un posible ataque a este algoritmo, lo que desencadena el desarrollo de nuevas variantes para el SHA. El

primero en esta familia es el SHA-1, el cual genera un hash de 20 bytes de un mensaje de hasta 2^{64} bits. Sin embargo, luego del ataque a MD5, se demostró también que SHA-1 tenia los días contados [22].

Las recomendaciones actuales [23] indican el uso de hashes mas fuertes que el SHA-1, como puede ser el SHA-256 o SHA-512, que generan hashes de 32 y 64 bytes respectivamente. La raíz está firmada desde Julio 2010 utilizando el hash SHA-256 encriptado con claves RSA con módulo de 2048 bits para las KSK y de 1024 para la ZSK.

7.2.2.1 - Tamaño de claves

Un aspecto fundamental en el proceso de firmado de una zona, es la elección del tamaño de las claves a utilizar. El administrador de la zona deberá tener en consideración cuantos datos −texto plano− será firmado y el periodo de publicación de las claves. El par de claves generado −público-privada− deberá ser lo suficientemente larga para prevenir que otros puedan obtener la parte privada usando criptoanálisis durante todo el periodo de validez de las claves. Como se mencionó anteriormente, la raíz utiliza claves de modulo 2048 bits para la KSK, por lo que no es recomendable utilizar claves más largas para dominios inferiores. La cadena de confianza es tan fuerte como la clave más débil. Por lo tanto, se recomienda −RFC4641− utilizar claves de 2048 para dominios de mayor nivel –TLD− y de 1024 para niveles inferiores. Sin embargo, esto esta dejado enteramente a decisión del administrador de la zona y este puede decidir entre cualquier largo estandarizado en DNSSec.

Con respecto al periodo de validez de las claves, está directamente asociado al uso que se le da a las mismas. Dado que la ZSK puede llegar a utilizarse de forma diaria está tiene más riesgo de ser comprometida por robo o pérdida, por lo que debe cambiarse cada menos tiempo que la KSK que solo se utiliza en los roll-over de las ZSK. Nuevamente, el administrador tiene que realizar un trade-off entre performance y seguridad. El cambio de clave −roll-over− muy seguido puede no llegar a ser necesario, pero dejarlas más tiempo de lo recomendado puede significar riesgos de seguridad. La ZSK de la raíz tiene un periodo de validez de 90 días, transcurrido ese tiempo se firmará la zona con una nueva clave y estará disponible un RR DNSKEY conteniendo su porción pública para realizar las validaciones. Diferentes procedimientos para cambiar las claves se verán más adelante.

7.3 - Procedimientos

Más allá de los procedimientos técnicos involucrados en la generación de estas claves, como la utilización de herramientas como DNSSec-keygen de Bind, queremos en esta sección detallar los procedimientos y medidas que el protocolo DNSSec recomienda en los aspectos envolventes de la generación de los pares de claves y su administración durante su vida útil. Estos procedimientos se encuentran documentados en [27] en forma general, y en [28] se describen en función de las necesidades de la zona raíz.

7.3.1 - Publicación de clave pública

Una de las principales preocupaciones de un administrador de una zona firmada, es como publicar las claves públicas de forma segura. DNSSec no tiene una PKI –Public Key Infrastructure− ni Autoridades Certificadoras –CA−, esto es debido a que los procedimientos de gestión de claves están basados en políticas locales. Para esto, el administrador de la zona deberá valerse de métodos propios de DNSSec –in band− y métodos fuera del protocolo –out of band−.

7.3.1.1 - A través de la zona padre firmada

En caso de que la zona padre esté firmada, el administrador de la zona debería proveer un mecanismo para la publicación de claves públicas de zonas hijas en su propia zona, tanto en forma de RR DNSKEY o DS, para ser utilizadas como SEP.

7.3.1.2 - DLV –DNSSec Look aside Validation−

ISC [24] ofrece una extensión a DNSSec –no es parte de este protocolo− que pretende ayudar en las etapas iniciales del despliegue de DNSSec, simplificando la configuración de los servidores recursivos –security-aware−. Es de esperar que este despliegue tome algunos años, lo que causará que se generen lo que ya denominamos como Islas de Seguridad. Si un servidor de nombres desea validar los datos provenientes de dichas zonas debe tener configuradas manualmente sus respectivos Secure Entry Point –SEP−, los cuales pueden llegar a ser demasiados, siendo una tarea inmanejable para la mayoría de los administradores de servidores. DLV ofrece un SEP adicional al del root, desde el cual se puede acceder al trust anchor –claves de confianza− de estas islas y así validar la información. Esto requerirá que las claves de confianza sean enviadas de forma segura al repositorio de DLV y los servidores de nombre configurados correctamente. Sin embargo, hasta el momento solo BIND ofrece la funcionalidad DLV.

7.3.1.3 - Fuera de banda –Out of Band−

Aparte de la publicación en la zona padre, no existe un método in band para la publicación de la primera clave pública que será utilizada como SEP –Trust Anchor−. Por lo que es importante, mientras no exista una cadena de confianza hasta la raíz, que los dominios importantes cuenten con mecanismos para dicha publicación. El más común es una web sobre SSL. Cualquier método que se elija debe asegurar integridad y autenticidad de la claves y ofrecer notificaciones cuando se actualizan las claves o las mismas son revocadas.

Los administradores de una zona deben siempre asumir que sus claves pueden ser utilizadas como SEP, aun si estas pertenecen a una cadena de confianza ya establecida, por lo que se recomienda mantener estos métodos fuera de banda junto con otros mecanismos in-band propios de DNSSec.

7.3.1.4 - In-Band

Los mecanismos que ofrece DNSSec para la publicación de claves públicas y sus renovaciones los veremos un poco más delante de forma detenida, sin embargo, se puede ir adelantando que estos se dividen en roll−over y Actualización de Trust Anchor. Pero estos mecanismos requieren la pre-existencia de claves ya verificadas que validen estas nuevas claves.

7.3.2 - Almacenamiento de clave privada

Dada la importancia de las claves privadas –como en cualquier sistema de encriptación asimétrica−, es necesario un medio de almacenamiento seguro, en especial para las KSK y claves de confianza. En operaciones normales de DNSSec, las claves no necesitan estar continuamente disponibles en los servidores autoritativos. Por el momento dejaremos de lado el caso de DDNS –Dynamic DNS− en donde esto no es cierto.

Las ZSK son necesarias en el momento del firmado de la zona, cuando los registros de la misma cambian y deben ser re-firmados o si las firmas expiran. Por otro lado, las KSK son necesarias para realizar los roll-over de las ZSK y las propias KSK o cuando la firmas del RRset DNSKEY firmado con la KSK expira. Por lo tanto, mientras estas no son utilizadas deben permanecer almacenadas y varias decisiones deben tomarse en función de las características de cada zona. Al elegir un medio adecuado de almacenamiento se debe realizar un compromiso entre el riesgo que estamos dispuestos a asumir y el costo. Estos costos consisten no solo en adquirir el medio físico de almacenamiento y software, sino también los costos de transporte de la clave hacia donde se realiza la firma –asumiendo que se encuentran en lugares separados− y el esfuerzo necesario para remediar casos de pérdida o si las claves son comprometidas. Y estos costos no son los mismos para la KSK como para la ZSK, por lo que procedimientos distintos deben planearse para cada tipo de clave.

Los niveles de riesgo a los que se enfrenta cada administrador están relacionados con el impacto que un ataque exitoso puede implicar. Las denominadas zonas de vanidad −vanity-zone−, “hojas” del árbol de delegación DNS, probablemente tendrán un nivel de riesgo muy bajo dado el poco efecto –a DNS o a los servicios que ofrece− que un ataque causaría. A medida que subimos por el árbol DNS, encontramos los TLD y la raíz, las cuales pueden provocar muchos daños si son comprometidas. También debe tenerse en cuenta las zonas que contienen información importante, como registros relacionados a una web de un banco, las cuales tienen más interés para un atacante.

Tomando lo anterior en consideración, se debe elegir de qué manera el equipo responsable de firmar la zona accede a las claves privadas. Puede ser de manera directa –online– o indirecta –offline– donde se requiere el uso de algún medio de almacenamiento físico como una smart card. Un punto intermedio entre estas dos, se conoce como "hidden master", en la que la computadora que realiza la firma es solo accesible por un canal seguro y solo a través de otros servidores maestros por medio de una consola o automáticamente.

Si se utiliza un mecanismo online, debería utilizarse un HSM –Hardware Security Module– para hacer uso de la clave. Estos dispositivos son utilizados para generar, almacenar y utilizar claves criptográficas. Es recomendable hacer uso de estos dispositivos ya que los mismos realizan todas las operaciones criptográficas de forma más rápida y segura. Estos dispositivos cuentan con sus propias protecciones físicas y eléctricas. El estándar FISP–140 [25] define los niveles de seguridad que proporcionan.

- Nivel 1: En nivel más bajo de seguridad. Un modulo de este nivel no precisa tener protecciones físicas y solo cuenta con un solo algoritmo o función incorporada.
- Nivel 2: El nivel 2 incorpora detección de adulteración o manipulación no autorizada y autentificación basada en roles.
- Nivel 3: Además de detección de adulteración en este nivel el HSM debe proveer resistencia a la adulteración y autentificación basada en identificaciones.
- Nivel 4: Este es el máximo nivel especificado en FIPS 140. Este nivel requiere una completa protección entorno al HSM, que detecte y reporte todo intento no autorizado y anomalías del entorno –cortes de energía, altas temperaturas, etc.–.

Para la generación de la KSK de la zona raíz y las operaciones y almacenamiento de la componente privada, ICANN utiliza un HSM nivel 4. Otro mecanismo de seguridad implementado por esta entidad para asegurar la clave privada, es la utilización del denominado control multi-persona, el cual consiste en dividir los datos necesarios para activar el HSM y hacer uso de la componente privada de la KSK en n partes y repartirla entre personas de confianza seleccionados por la comunidad de Internet y no pertenecientes a organizaciones relacionadas a la administración de la zona. Se define un umbral de smartcards del total que fueron creadas y repartidas para cada HSM, que son necesarias para hacer uso de la clave. Para activar la KSK, para firmar la zona por ejemplo, es necesario 3 de las 7 smartcards. También se utiliza un sistema similar para el backup de la clave privada KSK. Se almacena una copia encriptada de la clave en un dispositivo de almacenamiento portable, y la clave que se utilizó para la encriptación es distribuida utilizando el mismo esquema involucrando otros individuos de confianza, pero a diferencia del anterior procedimiento, aquí se necesitan 5 de 7 para recuperar la KSK.

7.3.3 - Key roll-over

Utilizaremos el término *roll-over*, heredado del Inglés, para hacer referencia al procedimiento de cambiar una clave –KSK o ZSK–. Los procesos involucrados en estas operaciones repercutirán otros servidores externos. En el caso del roll-over de la KSK los servidores autoritativos de la zona padre deberán actualizar el RR DS correspondiente, si dicha KSK es utilizada por algún servidor recursivo como clave de confianza, estos tendrán que actualizarla para poder seguir construyendo la cadena de confianza.

Criptográficamente las claves deben ser cambiadas en algún momento por las siguientes 3 razones:

- Un atacante experimentado puede llegar a acumular en un periodo de tiempo, suficiente texto plano y material encriptado como para realizar un análisis con el propósito de obtener la clave privada.
- Un ataque de fuerza bruta en algún momento dará resultados. Si las claves son cambiadas antes de que dicha amenaza sea factible, el atacante tiene que empezar de nuevo.
- Por alguna razón las claves pueden ser comprometidas, sin conocimiento de los usuarios u operadores de la zona, por lo que el atacante será capaz de modificar la información. Si bien no se puede hacer nada si se desconoce este hecho, cambiar las claves con frecuencia minimizaría los daños.

Cuando una clave es cambiada, dependiendo si se trata de la KSK o la ZSK, el impacto en los siguientes procesos será diferente y se debe tener en cuenta que los cambios que se deben realizar generalmente estarán controlados por otras entidades.

- Actualizar el registro DS en la zona padre: Si la zona padre en la cual se encuentra el RR DS que debe ser actualizado no es operado por el dueño de la zona hija, la sincronización entre el cambio de la KSK y su DS es muy difícil si no se tienen sistemas de automatización.
- Actualizar un ancla de confianza en un servidor security-aware: Este proceso depende del método usado, que veremos más adelante, pero en el peor caso podría consistir en actualizar manualmente los servidores, lo que podría tomar días. Durante ese lapso toda la información de la zona será reportada como no segura y rechazada por cualquier resolver que valide DNSSec. En un caso ideal, esto se aplicaría solo para la zona raíz, sin embargo, la existencia de islas de seguridad y múltiples anclas de confianza le dan una relevancia a este proceso durante la etapa de despliegue de DNSSec.
- Los RR RRSIG y DNSKEY almacenados en cache: Un resolver validador puede llegar a obtener estos registros en diferentes fechas, lo que podría causar que uno de ellos caduque antes. Si las claves almacenadas en los RR DNSKEY vencen después que los RRSIG, se rechazarán los datos firmados.

Visto lo anterior, parece difícil encontrar un momento adecuado para realizar el cambio de claves. Afortunadamente, el protocolo permite la existencia de múltiples claves –cada una definida es su respectivo RR DNSKEY − en la zona, y cada una de ellas debe ser probada para validar las firmas antes de rechazarlas, aunque una sola es necesaria para aprobarlas. Esta característica permite operar una zona firmada con las claves viejas y nuevas, lo que garantiza que los resolvers validadores hayan obtenido y validado las claves y firmas nuevas antes de que las antiguas hayan caducado. A continuación explicaremos los dos métodos que permiten el uso de múltiples claves: pre publicación y doble firmado.

7.3.3.1 - Pre publicación de claves

Este método permite que una o más claves sean incluidas en el RRset DNSKEY del ápex de la zona antes de comenzar a usarse activamente –una clave se dice activa si se utiliza para firmar registros−. Este método permite que las claves nuevas terminen siendo –en algún momento− almacenadas por todos los servidores recursivos, dejándolas disponibles en cache al momento de realizar el re-firmado con dicha clave. Este método es particularmente ventajoso en caso de que se tenga que realizar un roll−over de emergencia –cuando la clave privada se sospecha fue comprometida, por ejemplo−. El administrador de la zona será capaz de cambiar rápidamente de una clave a otra con un mínimo impacto, ya que le nueva clave ya se encuentra distribuida entre los resolvers. Otra ventaja importante, especialmente para zonas grandes, es que no se precisa firmar la zona con dos claves diferentes, lo que provocaría que el archivo de zona duplicara su tamaño.

Observemos ahora esto ilustrado en las siguientes etapas, donde realizaremos el roll-over de la ZSK:

Tabla 7-2: Roll-over ZSK, Pre publicación

Inicio	Nueva DNSKEY	Nuevo RRSIGs	Remover DNSKEY
SOA 1[4]	SOA 2	SOA 3	SOA 4
RRSIG SOA 1	RRSIG SOA 2	RRSIG SOA 3	RRSIG SOA 4
DNSKEY 1 (KSK)	DNSKEY 1 (KSK)	DNSKEY 1 (KSK)	DNSKEY 1 (KSK)
DNSKEY 2 (ZSK)	DNSKEY 2 (ZSK)	DNSKEY 2 (ZSK)	
	DNSKEY 3 (ZSK)	DNSKEY 3 (ZSK)	DNSKEY 3 (ZSK)
RRSIG 2 (DNSKEY 1)	RRSIG 2 (DNSKEY 1)	RRSIG 2 (DNSKEY 1)	RRSIG 2 (DNSKEY 1)
RRSIG 3 (DNSKEY 2)	RRSIG 3 (DNSKEY 2)	RRSIG 3 (DNSKEY 3)	RRSIG 3 (DNSKEY 3)

En el estado inicial tenemos la zona conteniendo la KSK y la ZSK la cual es utilizada activamente para firmar los RRset, incluyendo el RRset DNSKEY. El RRSIG 2 es la firma del RRset DNSKEY firmado por la KSK, mientras que el RRSIG 3 representa las firmas de todos los RRset de la zona. En una segunda etapa, se introduce la nueva ZSK pero no es utilizada para realizar ninguna firma. La duración mínima de esta etapa es el tiempo que le toma al nuevo registro propagarse por los resolvers validadores, más un

[4] Numero de serie de la zona. El cual se incrementa luego de cada cambio.

TTL[5]. Lo que garantiza que cuando se remuevan las firmas realizadas por la clave vieja en la tercera etapa, la nueva clave ya estará disponible para validar registros. Observar que al momento de incluir las nuevas firmas, las anteriores son removidas del archivo de zona. En una última etapa se retira la antigua clave. Es importante mantener esta clave por lo menos durante un TTL para asegurar que los resolvers que aún no han almacenado los nuevos RRSIGs puedan todavía validar los datos.

Se puede mejorar un poco este mismo esquema si se utiliza la etapa de remoción para introducir la futura clave que remplazará la nueva.

7.3.3.2 - Doble firmado

Así como lo indica el nombre del método, este consiste en firmar dos veces la zona con claves diferentes. Supongamos nuevamente que realizamos el roll-over de la ZSK, por lo que introduciremos un nuevo registro DNSKEY cuya clave firmará todo el contenido de la zona.

Tabla 7-3: Roll-over ZSK, Doble firmado

Inicio	Nueva DNSKEY	Remover DNSKEY
SOA 1	SOA 2	SOA 3
RRSIG SOA1	RRSIG SOA 2 (DNSKEY 2)	RRSIG SOA 3
	RRSIG SOA 2 (DNSKEY 3)	
DNSKEY 1 (KSK)	DNSKEY 1 (KSK)	DNSKEY 1 (KSK)
DNSKEY 2 (ZSK)	DNSKEY 2 (ZSK)	
	DNSKEY 3 (ZSK)	DNSKEY 3 (ZSK)
RRSIG 2 (DNSKEY 1)	RRSIG 2 (DNSKEY 1)	RRSIG 2 (DNSKEY 1)
RRSIG 3 (DNSKEY 2)	RRSIG 3 (DNSKEY 2)	
	RRSIG 4 (DNSKEY 3)	RRSIG 4 (DNSKEY 3)

A diferencia del método de pre publicación de clave, aquí tenemos un periodo de tiempo en el cual la zona contiene dos firmas por RRset, tres en el caso del RRset DNSKEY. En cualquier instante del proceso, las firmas nuevas pueden ser verificadas con las anteriores y viceversa. La duración de la etapa desde la introducción de la nueva clave y sus firmas hasta la remoción de las antiguas debe ser de hasta el máximo TTL de la zona, asegurando así que los caches no contienen las antiguas firmas al momento de remover la clave vieja.

7.3.3.3 - Ventajas y desventajas de estos métodos

El primer método, tiene como ventaja que no se firman los registros dos veces, lo que provocaría que se duplique por momentos el tamaño del archivo de zona, lo que puede traer complicaciones en zonas grandes como las TLD. Una pequeña desventaja de la pre publicación es que requiere 4 etapas y más trabajo del administrador de la zona padre si se trata del roll-over de la KSK.

[5] Se debe tomar como referencia el TTL más grande para la zona.

En general se utiliza y recomienda, realizar el roll-over de la KSK con el método de la doble firma, y la del ZSK con el método de pre publicación.

7.3.4 - Roll-over de la KSK

Existe una diferencia fundamental entre el roll-over de la KSK y la ZSK, y es que la primera requiere participación de la zona padre, mientras que en la segunda se puede operar independientemente del mismo. La ZSK permite, como ya hemos visto, tanto el mecanismo de doble firmado como el de pre publicación, esto es principalmente debido a que la KSK no cambia durante el roll-over de la ZSK, por lo que su copia almacenada en cache aún puede validar las nuevas claves. Sin embargo, aunque el método de pre publicación también funcionaria para la KSK, esta trae un problema que se denomina *clave tonta*. Analicemos un posible escenario para el roll-over de la KSK utilizando la pre publicación.

Tabla 7-4: Pre publicación KSK en zona padre

Inicio	Nuevo DS	Nuevo DNSKEY	Remover DNSKEY/DS
SOA 1	SOA 2	---------->	SOA 3
RRSIG SOA	RRSIG SOA	---------->	RRSIG SOA
DS 1	DS 1	---------->	
	DS 2	---------->	DS 2
RRSIG DS	RRSIG DS	---------->	RRSIG DS

Tabla 7-5: Pre publicación KSK en zona hija

Inicio	Nuevo DS	Nuevo DNSKEY	Remover DNSKEY/DS
SOA 1	---------->	SOA 2	---------->
RRSIG SOA	---------->	RRSIG SOA	---------->
DNSKEY 1 (KSK)	---------->		
		DNSKEY 3 (KSK)	---------->
DNSKEY 2 (ZSK)	---------->	DNSKEY 2 (ZSK)	---------->
RRSIG 2 (DNSKEY 1)	---------->	RRSIG 2 (DNSKEY 3)	---------->
RRSIG 3 (DNSKEY 2)	---------->	RRSIG 3 (DNSKEY 2)	---------->

Cuando la zona hija desea cambiar su KSK, debe notificar a la zona padre –etapa Nuevo DS en Tabla 7-4–, y enviarle el registro DS o también el DNSKEY correspondiente. Se publica en la zona padre los DS 1 y DS 2 que apuntan a la DNSKEY 1 y DNSKEY 2 respectivamente. Durante el roll-over –etapa Nuevo DNSKEY en Tabla 7-5–, que puede tomar lugar enseguida después el nuevo RR DS se haya propagado a través del DNS, en la zona hija se remplaza el DNSKEY 1 por el DNSKEY 2, e inmediatamente después de esto se puede notificar al padre para que retire los registros viejos de su zona –etapa Remover DNSKEY/DS–.

Este método en este caso presenta un inconveniente durante el tiempo en que el nuevo DS está presente en la zona padre y aun no se ha insertado su

correspondiente DNSKEY. La zona padre, no puede verificar la correspondencia entre ambos registros, y además este nuevo DS que no apunta a ninguna clave publicada se lo conoce como clave tonta.

La segunda alternativa, y la preferida para este caso, es realizar el doble firmado para el roll-over de la KSK. A diferencia de la ZSK, aquí no está el problema de duplicación del archivo de zona ya que la KSK solo firma las claves. Estudiemos ahora los estados que involucra este procedimiento.

Tabla 7-6: Doble firmado KSK en zona padre

Inicio	Nuevo DNSKEY	Nuevo DS	Remover DNSKEY
SOA 1	---------->	SOA 2	---------->
RRSIG SOA	---------->	RRSIG SOA	---------->
DS 1	---------->	DS 2	---------->
RRSIG DS	---------->	RRSIG DS	---------->

Tabla 7-7: Doble firmado KSK en zona hija

Inicio	Nuevo DNSKEY	Nuevo DS	Remover DNSKEY
SOA 1	SOA 2	---------->	SOA 3
RRSIG SOA	RRSIG SOA	---------->	RRSIG SOA
DNSKEY 1 (KSK)	DNSKEY 1 (KSK)	---------->	
	DNSKEY 3 (KSK)	---------->	DNSKEY 3 (KSK)
DNSKEY 2 (ZSK)	DNSKEY 2 (ZSK)	---------->	DNSKEY 2 (ZSK)
	RRSIG 1 (DNSKEY 1)	---------->	
RRSIG 2 (DNSKEY 1)	RRSIG 2 (DNSKEY 2)	---------->	RRSIG 2 (DNSKEY 2)
RRSIG 3 (DNSKEY 2)	RRSIG 3 (DNSKEY 3)	---------->	RRSIG 3 (DNSKEY 3)

En este estado inicial tenemos una delegación segura con DS 1 y DNSKEY 1. La zona hija quien inicia el roll-over genera una nueva clave KSK, la publica en su zona y firma los RRset DNSKEY con la misma. Tanto la clave –pública– como el RR DS son proporcionados al administrador de la zona padre quien puede verificar que los registros que recibió corresponden con los presentes en su zona hija. Luego de propagarse el nuevo DS por todos los servidores autoritativos de la zona padre, se debe esperar por lo menos un TTL –del RR DS– pasa asegurarse de que los caches no contengan registros viejos al momento de retirar la KSK vieja.

Tener en cuenta que la responsabilidad de mantener una cadena de confianza valida recae en el administrador de la zona hija. Aunque es el administrador de la zona padre quien debe proveer mecanismos seguros y fuera de DNS para asegurar dicho intercambio.

7.3.5 - Roll-over de Algoritmo

El creciente poder computacional hace necesario considerar la eventualidad de que en algún momento se deberá cambiar el algoritmo utilizado para generar las claves y las firmas digitales. DNSSec establece que debe existir un RRSIG por cada RRset usando por lo menos un DNSKEY de cada algoritmo definido dentro del RRset DNSKEY en el ápex de la zona. Si bien esto no introduce ningún problema al momento del roll-over si el algoritmo ya está incluido en el RRset DNSSKEY, sí trae complicaciones cuando el roll-over involucra un cambio de algoritmo.

Supongamos el siguiente escenario: Un cache tiene almacenado las claves y firmas viejas –que deben ser cambiadas– y se introducen las claves nuevas usando otro algoritmo. Supongamos que las claves expiran antes que las firmas, por lo que el cache deberá obtener las nuevas, y como las firmas aun no caducaron no precisas obtener las nuevas. Por lo que el cache tiene almacenado una clave con un algoritmo que no corresponde con ninguna firma presente en el cache, lo que resulta en que al validar se rechacen las firmas. De forma análoga tenemos el mismo problema si las firmas expiran antes que las claves.

Para solucionar este problema se plantea el siguiente esquema [26] que incluye unos pasos adicionales a los roll-over vistos anteriormente.

- Al momento de incluir un nuevo algoritmo, agregar primero a la zona las firmas, y luego de que todas las firmas antiguas hayan expirado de cache, agregar las claves.
- Para remover un algoritmo, quitar primero los DNSSKEY que lo utilicen dejando las firmas, y remover las firmas viejas cuando estas expiren de los caches.

Tabla 7-8: Roll-over de algoritmo

Inicio	Nuevos RRSIG	Nuevo DNSKEY	Rem. DNSKEY	Remover RRSIG
SOA 1	SOA 2	SOA 3	SOA 4	SOA 5
RRSIG SOA 1	RRSIG1 SOA 2	RRSIG1 SOA 3	RRSIG1 SOA 4	
	RRSIG2 SOA 2	RRSIG2 SOA 3	RRSIG2 SOA 4	RRSIG2 SOA 4
DNSKEY 1 (ZSK)	DNSKEY 1	DNSKEY 1		
RRSIG1 (RRset)	RRSIG1 (RRset)	DNSKEY 2	DNSKEY 2	DNSKEY 2
	RRSIG2 (RRset)	RRSIG1 (RRset)	RRSIG1 (RRset)	
		RRSIG2 (RRset)	RRSIG2 (RRset)	RRSIG2 (RRset)

En el paso inicial tenemos las firmas genéricas RRSIG1 que utilizan la clave DNSKEY 1 a la cual se quiere realizar un roll-over de algoritmo. En el segundo paso incluimos los nuevos RRSIG dejando de lado la DNSKEY 2 para el tercer paso, en el cual se retiran las claves con el algoritmo viejo. Por último se eliminan los RRSIG de las claves anteriormente removidas. De esta manera nos aseguramos que la integridad criptográfica de la zona no es comprometida por el roll-over del algoritmo.

7.4 - Actualización automática de las Trust Anchors

Como ya hemos mencionado, el escenario final ideal del despliegue de DNSSec consistiría en que todas las zonas estén firmadas y que no existan islas de seguridad, y que con la clave de confianza o trust anchor de la raíz sea posible validar todas las delegaciones y por lo tanto los datos DNS. Ahora consideremos el problema de actualizar esta clave de confianza almacenada en todos los resolvers que la utilicen, ya sea porque su tiempo de vida expiró o por razones de seguridad –la clave se vio comprometida–. Este problema es diferente al roll-over de la KSK, ya que se efectúa en los resolver y no en los servidores autoritativos como ha sido hasta ahora.

Un resolver esta configurado para aceptar determinadas claves como trust anchor y utilizar la zona que firmó como SEP. En BIND esto se puede hacer de forma manual de la siguiente manera:

Ejemplo 7-1: Configuración de clave aceptada

```
trusted-keys {

example.com. 257 3 5 "Aalskdfjwlr3lkjaldjfalk3raljdfal...";
}
```

Una clave de confianza no es otra cosa que una copia del RR DNSKEY de la zona que utilizaremos como SEP, la única diferencia con otras KSK es que sabemos que es de confianza. Nuestro problema ahora será cómo actualizarlas de forma de conservar siempre una clave de confianza en los resolvers y nunca dejar de ser operativos.

Tomaremos como punto de partida que una clave de confianza inicial ya fue configurada en todos los resolver validadores usando algún mecanismo fuera de banda. A partir de este escenario, describiremos un mecanismo propuesto en la RFC 5011 [27] por el cual un resolver podrá actualizar un trust anchor para cierto SEP, sin intervención humana a nivel del resolver. Existen por supuesto casos especiales que escapan el alcance de la propuesta, como la pérdida total de claves de confianza, lo que significaría la configuración manual de nuevas claves en el resolver.

7.4.1 - Teoría de operación

La idea principal de este método es que las claves de confianza existentes en los resolvers puedan ser usadas para autentificar nuevas claves de confianza para cada SEP. Cuando el operador de una zona agrega una nueva clave de confianza –una KSK– y el resolver puede validarla usando la clave de confianza ya configurada en el mismo, esta nueva clave podrá ser usada como trust anchor para ese SEP.

Un tema igualmente importante es como eliminar claves de confianza que han sido comprometidas. Una clave comprometida puede ser utilizada para agregar datos falsos a la zona y también revocar las claves no comprometidas. Por lo que los mecanismos de revocación de claves deben ser capaces de revocar una clave aun cuando esta ha sido comprometida.

7.4.2 - Revocación de claves de confianza

Asumamos que tenemos dos trust anchor configuradas en un resolver, A y B, y sabemos que la clave B fue comprometida. Para evitar que B anule a A, simplemente mandando una firma de RR DNSKEY que no contenga a A, el método propuesto aquí [27] requiere conocimiento de la parte privada de la clave para revocar su DNSKEY.

Una clave se considerará revocada cuando el resolver vea la clave que firmo su propio RRset y la clave tiene seteado el bit REVOKE. Desde ese momento, el resolver no utilizará esa clave para validar ningún registro, salvo el RRSIG utilizado para validar la revocación. A diferencia del procedimiento de agregar una clave al resolver, la revocación es de carácter inmediato y permanente una vez validada la revocación. De esta forma aunque la clave B este comprometida, un atacante no podrá revocar la clave A.

7.4.3 - Hold-down

Veremos ahora como agregar una clave nueva en los resolvers para cierto SEP. Sigamos con el ejemplo de que B ha sido comprometida, y el atacante pretende agregar una nueva clave C como trust anchor –adicionándola al RRset DNSKEY y firmándolo con B–, lo que implicaría que tanto el operador de la zona como el atacante pueden agregar datos validos a la zona. Para evitar esa posibilidad, se implementa lo que se denomina un tiempo de hold-down para la validación del trust anchor. Cuando un resolver ve una nueva trust anchor en una zona de confianza –una zona de la cual el resolver posee una trust anchor y utiliza como SEP–, inicia un timer para dicha clave. Mientras el resolver vea dicha clave y la pueda validar mantendrá activo dicho timer, si en algún momento la clave no se encuentra más en un RRset validado, el timer se resetea. Lo mismo sucederá si las claves utilizadas para validar la nueva trust anchor son revocadas antes de que expire el timer, este se resetea y el proceso de aceptación de una nueva trust anchor debe comenzar de nuevo.

En caso de que el timer expire antes de cualquier acontecimiento descrito anteriormente, el resolver debe nuevamente obtener las claves del servidor autoritativo y validarlas. A partir de ese momento, si es que esta ultima validación es correcta, la nueva trust anchor será considerada segura y podrá utilizarse para validar nuevas claves y datos de la zona SEP. El tiempo máximo que una clave precisa para volverse una trust anchor será entonces la suma del tiempo hold-down más un TTL del RR DNSKEY de la misma.

De esta manera podemos evitar que el atacante que posee la clave comprometida B, pueda agregar una nueva clave C. Revocando B y firmando el RRset DNSKEY con A y B, el contador de C se reseteará y no podrá volver a ser iniciado por la clave B ya que esta fue revocada.

Es importante notar que la nueva clave no tiene por qué estar siempre en el resolver, debido al TTL, el DNSKEY de dicha clave puede expirar por lo que el cache necesita obtenerlo de nuevo del servidor autoritativo. Lo que no significa que el timer

deba resetearse. Esto se hará en caso de que la firma cubriendo los DNSKEY no contiene a la clave nueva.

7.4.4 - Refresh continuo

Un resolver que está configurado para actualizar de forma automática las trust anchor para determinada zona de confianza, debe preguntar por la misma clave y nuevas para dicha zona –realizando una consulta por el DNSKEY y sus RRSIG correspondiente–. Este "refresh" debe realizarse no más frecuentemente que una vez por hora, y no menos frecuente que el mínimo de {15 días, ½ TTL original, ½ del intervalo de expiración del RRSIG}. Siendo el intervalo de expiración del RRSIG el tiempo desde que el registro es obtenido hasta que expira la firma.

7.4.5 - Parámetros para el resolver

7.4.5.1 - Tiempo de Hold-down

El tiempo de hold-down, definido para agregar una nueva trust anchor, es de 30 días o el TTL del primer RRset DNSKEY que contenga la nueva clave, el que sea mayor de los dos tiempos. De esta manera el resolver se asegura de ver por lo menos 2 RRsets DNSKEY que contienen la nueva clave antes de aceptarla.

7.4.5.2 - Tiempo de Hold–down para remover claves

Este tiempo declarado en los resolvers, es puramente arbitrario, y es simplemente para definir cuanto tiempo, luego de revocada o expirada una clave, permanecerá almacenada en el cache. En general está definido como 30 días, pero este parámetro no impacta para nada en la seguridad del protocolo, sino que evita que el resolver se vea abarrotado de información desactualizada.

7.4.5.3 - Trust anchor por zonas de confianza

Este parámetro es más una referencia para los diseñadores de software para que tengan en cuenta que un resolver debe ser capaz de soportar hasta 5 claves de confianza por zona definida como SEP.

7.4.6 - Tabla de estados

La actualización de claves de confianza tiene lugar en los resolvers, por lo que es importante comprender y realizar el análisis de lo que el resolver ve y puede hacer con la información que tiene en cada momento. A continuación expondremos una

tabla de estados que muestra las diferentes decisiones del resolver a lo largo de la vida de una clave de confianza.

La columna de la izquierda indica el estado actual, mientras que la fila superior muestra el próximo estado. Entre ambos se encuentran los eventos que desencadenan los cambios de estados.

Tabla 7-9: Estados del resolver. Actualización automática de Trusted Anchor

	Próximo Estado					
Actual	Start	AddPend	Valid	Missing	Revoked	Removed
Start		NewKey				
AddPend	KeyRem		AddTime			
Valid				KeyRem	RevBit	
Missing			KeyPres		RevBit	
Revoked						RemTime
Removed						

Estados:

- Start: La clave aun no ha sido vista por el resolver, o ha sido vista una vez y luego estuvo ausente en el siguiente RRset DNSKEY.
- AddPend: El resolver ve una nueva clave KSK y puede validar el RRset DNSKEY que la contiene utilizando la actual clave de confianza.
- Valid: La clave fue vista y validada en todos los RRset DNSKEY desde que fue vista por primera vez hasta el tiempo de hold-down, incluyendo la validación de una nueva firma obtenida de servidor autoritativo luego de expirado el timer. Recordar que la clave no tiene por qué estar presente en el servidor continuamente, debido a su TTL.
- Missing: Este estado se da únicamente cuando estando en el estado Valid, el resolver deja de ver la clave de confianza ya validada en el último RRset DNSKEY. Es recomendable no pasar por este estado, ya que el operador de la zona debería revocar correctamente la clave antes de eliminarla de la zona.
- Revoked: Cuando el resolver ve un RRSIG firmado por una clave con el bit REVOKE seteado pasa a este estado y dicha clave es inmediatamente considerada inválida para la autentificación de datos.
- Removed: Luego de un tiempo considerablemente largo, 30 días, luego de ser revocada una clave puede ya ser removida del resolver.

Eventos:

- NewKey: El resolver ve una clave KSK para cierta zona de confianza.
- KeyPres: La clave retorno a un RRset DNSKEY valido.
- KeyRem: El resolver ve un RRset DNSKEY que no contiene la clave nueva.

- AddTime: La clave fue vista por el resolver durante todo el tiempo de hold-down y validó un nuevo RRSIG cubriendo el RRset DNSKEY conteniendo la nueva clave.
- RemTime: Una clave revocada no apareció durante un tiempo prolongado en el RRset DNSKEY de la zona de confianza.
- RevBit: La clave es vista con el bit REVOKE seteado, y existe una firma cubriendo un RRset DNSKEY firmado por dicha clave.

7.4.7 - Eliminar una zona de confianza

Una zona de confianza, o SEP, de la cual el resolver ya no posee ninguna clave de confianza, debe considerarse como si nunca hubiese sido una zona de confianza. Si tampoco existe una clave de confianza superior –en la jerarquía DNS− a dicha zona, los datos de la zona e inferiores a esta deben considerarse inseguros. Si una clave de confianza superior existe debe existir una cadena de confianza –delegación segura− a esta zona para considerar los datos seguros nuevamente.

7.5 - Operaciones periódicas y de emergencia.

7.5.1 - El tiempo en DNSSec

El concepto de tiempo antes de DNSSec era relativo. El registro SOA como ya sabemos contiene campos como REFRESH, RETRY y EXPIRATION que son utilizados como timers por servidores esclavos, para mantenerse sincronizado con el servidor autoritativo maestro. El conocido TTL que todos los RR poseen es otro timer que tiene DNS para determinar cuánto tiempo un registro debe permanecer en cache.

Al hacer uso de firmas digitales, DNSSec introduce el concepto de tiempo absoluto en DNS. El periodo de validez de una firma es indicada por una fecha de inicio y una de expiración, después de la cual la firma es considerada inválida y por lo tanto sus datos también.

En la RFC 4641 se recomiendan ciertas precauciones a tomar en cuenta en relación a la expiración de las firmas y a los TTL de la zona. Se denominará como TTL de una zona, al máximo TTL que se puede encontrar en la zona –Maximum Zone TTL−, este parámetro se utilizará como referencia temporal.

En primer lugar el TTL de la zona debería ser una fracción del periodo de validez de las firmas. De lo contrario, si esta es del mismo orden o igual a dicho periodo de validez, entonces todas –o la gran mayoría− de las firmas serían almacenadas en cache hasta que estas expiren. Momento en el cual los resolvers podrían de forma simultánea solicitar nuevos registros, generando así un pico de carga en los servidores autoritativos cada vez que se vencen las firmas. Para evitar este tipo de

comportamiento, se sugiere que el TTL de zona sea algunas veces menor que el periodo de validez de las firmas.

Segundo, el periodo de publicación de una firma debería finalizar por lo menos un TTL de zona antes de la expiración de las firmas. En el peor de los casos, refirmar la zona instantes antes de la expiración de las firmas causaría un pico de tráfico ya que todos los resolvers estarían solicitando nuevos registros al mismo tiempo. Con esta precaución nos aseguramos que los resolvers puedan adquirir las firmas en un intervalo de tiempo más prolongado evitando sobrecargas de tráfico.

Se sugiere también que el TTL de la zona sea lo suficientemente grande como para poder obtener los registros y sus firmas así como también para verificarlas en la cadena de confianza. Un resolver validador deberá poder durante todo el tiempo que le lleve validar las firmas contar con los registros –RRset, DS, DNSEKY y RRSIGs– necesarios para realizarlas. Más específicamente, los registros en puntos de delegación como los DS y DNSKEY deberían tener TTL superiores de modo de evitar que verificaciones frecuentes sobrecarguen a los servidores recursivos.

Por último, debemos considerar que los timers del SOA también deben tomar en cuenta la fecha de expiración de las firmas digitales. Un servidor esclavo necesitará conseguir las firmas de la zona nuevas antes que las anteriores expiren con suficiente anticipación. Si el esclavo se encuentra desincronizado con el maestro –debido a que se perdió una conexión por ejemplo– este dará por expirada la zona cuando el timer EXPIRE del SOA así lo determine. En ese momento el servidor puede responder con un SERVFAIL o no emitir un flag AA –Respuesta Autoritativa–. Dado que no existe ninguna relación entre este timer y el periodo de expiración de las firmas, podría suceder que las firmas expiren mucho antes que el timer EXPIRE del SOA. Se sugiere entonces que el timer de expiración del registro SOA sea aproximadamente un tercio del periodo de validez de las firmas. Se considera que este tiempo permitirá identificar los problemas con las transferencias de zona antes de que expiren las firmas.

7.5.2 - Calendario de Roll-over

Habiendo visto ya todos los mecanismos necesarios para la realización de los roll-overs, resta ahora determinar cuándo usarlos. Según mencionamos en 7.3.3 - existen varias razones que desencadenan un cambio de clave, aquí los agruparemos en dos categorías: administrativos y de emergencia. Todo buen administrador de zona deberá generar un calendario que regirá los roll-over administrativos de la KSK y de la ZSK. Según las especificaciones actuales para la raíz [28], su KSK se debería cambiar cada 2 a 5 años como máximo, mientras que la ZSK tiene un ciclo de 90 días. Dado que estos calendarios se rigen en vigor de la vida útil de estas claves y sus algoritmos, es razonable que un administrador de zona siga los mismos delineamientos, más aún si utiliza el mismo algoritmo y módulo que la raíz. Veremos ahora un calendario modelo que utiliza las recomendaciones anteriores para el roll-over de la ZSK utilizando pre-publicación y el de la KSK utilizando doble firmado.

Tabla 7-10: Calendario Roll-Over

T−10	T	T+10	T+20	T+30	T+40	T+50	T+60	T+70	T+80	T+90
ZSK 1	ZSK 1									
ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2	ZSK 2
									ZSK 3	ZSK 3
KSK 1	KSK 1	KSK 1	KSK 1	KSK 1	KSK 1	KSK 1	KSK 1	KSK 1		
		KSK n	KSK n	KSK n	KSK n	KSK n	KSK n	KSK n	KSK n	KSK n

En T−10 días se publica la ZSK 2 de forma inactiva –no se utiliza para firmar registros−, y diez días después se remueve la ZSK 1 y se refirma la zona con la nueva ZSK. Este paso puede ser realizado en cualquier momento luego de ser insertada la nueva ZSK en la zona, pero es recomendable tener en cuenta el TTL del respectivo RR DNSKEY para hacerlo cuando todos los servidores recursivos tengan almacenado dicho registro de modo de minimizar la sobrecarga de tráfico durante el roll-over. La ZSK tendrá una vida útil de 90 días, hasta T+90, al finalizar este periodo una nueva ZSK entrará en vigencia comenzando así un nuevo ciclo. Es recomendable también que el operador de zona no realice el re firmado de zona de forma perfectamente periódica, sino que debe introducir algún elemento de aleatoriedad para evitar que un atacante utilice esos momentos para realizar un ataque DDoS, ya que cuando vencen las firmas se producen picos de tráfico.

En la Tabla 7-10 también incluimos el roll-over de la KSK, la cual en condiciones normales se puede realizar cada 2 años o más. La única conveniencia de realizar el roll-over de la KSK como se muestra en la tabla, es que se minimiza la cantidad de registros, claves y firmas presentes en la zona. Como vemos en la tabla anterior, en ningún momento hay más 3 DNSKEY presentes en la zona.

Según ya habíamos visto en 7.3.4 - existe un tiempo mínimo, determinado por el retardo de propagación de registros en DNS, durante el cual la KSK n y la KSK n-1 deben permanecer juntas en la zona. Del mismo modo, según analizamos en 7.4.3 - , existe un tiempo denominado hold-down, durante el cual una clave KSK utilizada como SEP –clave de confianza− debe estar presente en el RRset DNSKEY –y validarse – para poder considerarse también una clave de confianza.

8. Maqueta

Un estudio de DNSSec no hubiera estado completo sin una puesta en práctica de las operaciones y procedimientos descritos en los capítulos anteriores. En particular, nuestro interés estuvo centrado en estudiar el impacto de la implementación de DNSSec en servidores de nombres autoritativos y recursivos. Para esto se montaron diferentes maquetas, cada una representando el tipo de escenario que queríamos estudiar. Consideramos importante realizar las pruebas con datos reales, y para ellos el Servicio Central de Informática de la Universidad de la República −SeCIU− nos brindó su colaboración dándonos acceso a las trazas de tráfico de sus servidores DNS, junto con las configuraciones de los mismos y las zonas para las cuales SeCIU es autoritativo.

Las pruebas que se propusieron tuvieron como objetivo medir y comparar el desempeño de nuestro servidor autoritativo y recursivo, al operar con zonas firmadas y sin firmar. Debido al incremento en la cantidad de registros que se intercambian y el costo de CPU derivado de la verificación de firmas, esperamos encontrar diferencias de performance entre antes y después de firmar. Se debe aclarar que las pruebas se realizaron bajo la hipótesis de que todas las consultas están firmadas. Este es el peor caso posible respecto a la performance de los servidores, pero un caso ideal en el despliegue definitivo de DNSSec.

Clasificamos las pruebas a realizar de la siguiente manera:

1. Replica servidor SeCIU.
2. Autoritativo: Trafico UDP sobre un servidor autoritativo no recursivo.
3. Recursivo sin cache: Todas las consultas requieren recursividad para resolverse.
4. Recursivo con cache: El 70% de las consultas ya están almacenadas en cache.
5. Recursivo con delay: Las respuestas autoritativas traen un delay.
6. Autenticación de no existencia en servidores recursivos: Comparación entre NSEC y NSEC3.

8.1 - Maqueta y herramientas

Con el propósito de obtener medidas comparables, todas las pruebas realizadas tuvieron como actor principal a la misma computadora, a la que llamamos UbuntuDesktop. En las pruebas sobre el servidor autoritativo, el actor principal, UbuntuDesktop, cumplió el rol servidor autoritativo. Por otro lado, en las pruebas sobre servidores recursivos, esta computadora actuó en el papel de servidor recursivo. La topología de cada escenario la describiremos en cada parte, aquí nos detendremos a especificar los equipos utilizados y las herramientas escogidas para analizar los datos. Identificaremos los equipos utilizados con los siguientes nombres:

- UbuntuDesktop: Intel P4, 512 RAM, 1.6 GHz, OS: Ubuntu Desktop 10.04.
- UbuntuServer: Intel P4, 512 RAM, 1.6 GHz, OS: Ubuntu Desktop 10.04.
- ServerZona: i5, ACER 5740 2,27GHz 4GB RAM, OS Ubuntu Desktop 11.10.
- Mirror: i5 Dell INSPIRON N5010 2,4GHz 6GB RAM, OS Ubuntu Desktop 11.10.
- Cliente: i5 Dell INSPIRON N5010 2,4GHz 4GB RAM, OS Ubuntu Desktop 11.10.
- Switch: 3COM Super Stack II 3300.

Los equipos utilizados como servidores en las pruebas, UbuntuServer y UbuntuDesktop, son máquinas que además de tener unos cuantos años, no fueron diseñadas para actuar como servidores DNS. Por lo que no cuentan con la capacidad de servidores dedicados que se utilizan para servir zonas en operación. Sin embargo, veremos que fueron suficientes para la realización de nuestro proyecto.

Para generar las consultas necesarias en cada prueba utilizamos un programa escrito en C, el cual genera las consultas requeridas según las necesidades de cada escenario, enviándolas a la dirección de loopback de la maquina local −127.0.0.1, localhost−. Estas consultas fueron capturadas con el programa Wireshark [29] y luego modificadas utilizando la herramienta tcprewrite −perteneciente a la suite de programas de tcpreplay [30]−, de modo de modificarles las direcciones IP, MAC y el checksum para que pudieran adaptarse a las direcciones de los equipos de nuestra maqueta. Finalmente, recreamos el tráfico utilizando el programa tcpreplay [30], el cual permite reenviar el tráfico, previamente preparado, especificando los paquetes por segundo −pps−, cantidad total de paquetes a enviar, y una variedad más de parámetros.

Ejemplo 8-1: Uso de tcpreplay y tcprewrite

```
tcprewrite      --snet-smac=MAC_Cliente      --snet-dmac=MAC_Servidor      --
ipsrcmap=164.73.0.0/16:164.73.45.146   --ipdstmap=Server_SeCIU/32:164.73.45.145
-C -i inputfile -o outputfile

tcpreplay --pps=PPS --limit=LIMITE_PKT --intf1=eth0   outputfile
```

En el Ejemplo 8-1 se muestra el uso de tcprewrite para modificar el archivo *inputfile,* a éste se le modifican las direcciones IP y MAC, se recalcula el checksum del paquete y se guarda en el archivo *outputfile.* El segundo comando utiliza el archivo creado anteriormente y lo replica en la interfaz *eth0* a determinada velocidad medida en paquetes por segundo. El parámetro *LIMITE_PKT* especifica la cantidad de paquetes que se deben enviar, del total contenido en el archivo de entrada. Más información acerca del uso de estas herramientas se puede encontrar en [30].

Todas las computadoras utilizaron como servidor de nombres el programa BIND 9 del Internet Software Consortium −ISC [24]−. Si bien de hecho existen otras alternativas en programas de servidores de nombre, como OpenDNSSec [31], NSD [32] y Unbound [33] entre otros, elegimos BIND por ser la más difundida y elegida por administradores de zona, además de contar con mayor soporte técnico en la web.

Otra de las tareas que efectúa un servidor de nombres, además de responder consultas, es llevar un registro de lo que hace. Todo buen administrador de una zona sabe por experiencia que el debugging de problemas comienza observando los log del sistema, en particular el log de BIND. Para nuestras pruebas dispusimos que BIND realizara logs de las consultas entrantes y de todas las operaciones propias a DNSSec que realiza. Esto se logra mediante la inclusión del siguiente fragmento en el archivo de configuración named.conf.

Ejemplo 8-2: Configuración log de BIND

```
logging {
    channel query.log {
        file "/etc/bind/log/query.log" versions 5 5m;
       print-time yes;
        // Set the severity to dynamic to see all the debug messages.
        severity debug 3;
    };

    category queries { query.log; };
    category dnssec { query.log; };

};
```

Este fragmento del archivo de configuración del BIND, indica que se genera un archivo de log llamado query.log que contiene registros de las categorías señaladas, limitando el tamaño del archivo a 5 megabytes. Al llegar a ese tamaño se abrirá un nuevo archivo −con el mismo nombre−, pero conservando y sobrescribiendo de forma cíclica las 5 versiones anteriores de ese archivo. Con esto logramos mantener un log de las actividades del servidor y así asemejarnos al comportamiento normal de un servidor de nombres como el de SeCIU.

Una de las medidas más importantes que se realizó en todas las pruebas fue el consumo de CPU por parte del servidor de nombres de BIND, llamado *named*, y el tiempo IDLE −tiempo ocioso− del CPU. Para poder realizar estas medidas contamos con la suite de programas *sysstat* [34] de Linux, el cual contiene la aplicación *sar* y *pidstat.* La primera brinda estadísticas generales de CPU, entre ellas el tiempo CPU IDLE, y la segunda muestra los recursos utilizados por otro proceso identificado por su PID −Process ID−, en este caso el ID del named.

8.1.1 - Generación de claves y firmas

Como vimos en la sección 7. , es necesaria la utilización de claves públicas y privadas para operar una zona firmada. Estas deben ser generadas por el administrador de zona, el cual debe asegurarse que la porción privada permanezca segura y que la clave pública esté disponible en los servidores recursivos validadores. En nuestras pruebas utilizamos tanto las zonas de autoridad de SeCIU como zonas creadas por nosotros, y para cada una de ellas generamos dos pares de claves, un par correspondiente a la KSK y otro a la ZSK.

Siguiendo las recomendaciones planteadas en [23], elegimos los siguientes algoritmos para las firmas digitales:

- KSK: RSA con módulo de 2048 bytes, Hash SHA 256.
- ZSK: RSA con módulo de 1024 bytes, Hash SHA 256.

En el caso de las pruebas de performance de NSEC3 las firmas deben utilizar SHA1, y las claves se identifican de forma diferente para mantener compatibilidad hacia atrás, ver Tabla 7-1.

Para la generación de estas claves utilizamos la herramienta *dnssec-keygen* incluida en el paquete BIND.

Ejemplo 8-3: Uso de dnssec-keygen

```
dnssec-keygen -a RSASHA256 -b 2048 -f KSK -n ZONE nombre_zona
```

El comando del Ejemplo 8-3 es utilizado para generar una clave KSK para la zona "*nombre_zona*" con las especificaciones dadas anteriormente. Este programa devuelve dos archivos, uno conteniendo la clave pública y otro la privada, con los siguientes nombres respectivos:

Ejemplo 8-4: Archivos de claves DNSSec

```
Knombre_zona.+alg_tag+key_tag.key
Knombre_zona.+alg_tag+key_tag.private
```

En la Tabla 7-1 se encuentran los posibles valores que puede tomar *alg_tag* en función del algoritmo utilizado. El *key_tag* se genera junto con la clave, y sirve para diferenciar e identificar claves del mismo dominio en una zona. El primer archivo contiene el RR DNSSKEY de la clave, el cual es insertado en el archivo de zona durante el proceso de firmado. El segundo archivo contiene la porción privada de la clave, y es utilizado para generar las firmas digitales.

El tiempo que toma generar una clave puede llegar a ser superior a una hora, dependiendo del equipo utilizado y la cantidad de entropía que este tenga almacenado. La entropía se puede interpretar como una medida de aleatoriedad; más entropía significa que la fuente de aleatoriedad del equipo devolverá símbolos −bytes−

más difíciles de predecir. Esto se traduce en que la calidad de las claves −dificultad en romperlas− generadas dependerá de la entropía que el equipo tenga disponible. En Linux se utiliza por defecto el archivo /dev/random como fuente de bits aleatorios. Éste obtiene su entropía de dispositivos que interactúen en el exterior, como el teclado, el mouse o la interface de red. También es posible utilizar hardware dedicado a la generación de claves, los cuales integran generadores de números aleatorios de mejor calidad y de forma más rápida.

Es posible disminuir estos tiempos con el propósito de generar claves de forma más rápida, pero a expensas de claves más débiles, especificando que en la generación de claves se utilice como fuente de bits aleatorios el archivo /dev/urandom. A diferencia de la fuente /dev/random, ésta utiliza un generador de números pseudoaleatorios inicializado utilizando entropía de /dev/random para generar símbolos de forma más rápida, pero con menos entropía; resultando en tiempos de generación más cortos pero claves más predecibles −menos seguras−. Es importante resaltar que este tipo de mecanismos no debe utilizarse para firmar zonas en producción, ya que debilita considerablemente la seguridad de los datos y de las claves. Sin embargo, para experimentar y realizar pruebas en un ambiente controlado, hacer uso de claves débiles no afectará los resultados obtenidos, ya que los algoritmos para encriptar y des-encriptar firmas no distingue entre claves débiles y fuertes.

Con las claves ya generadas y las zonas prontas, se procede a realizar las firmas digitales. Para esto, el paquete BIND brinda la herramienta *dnssec-signzone,* la cual toma como entrada el archivo de zona, las claves generadas y otros parámetros opcionales, devolviendo el archivo de zona que incluye todas las firmas y un archivo conteniendo los registros DS que deben incluirse en la zona padre.

Ejemplo 8-5: Uso de dnssec-signzone

```
dnssec-signzone -K directorio_claves –o origen  zonefile
```

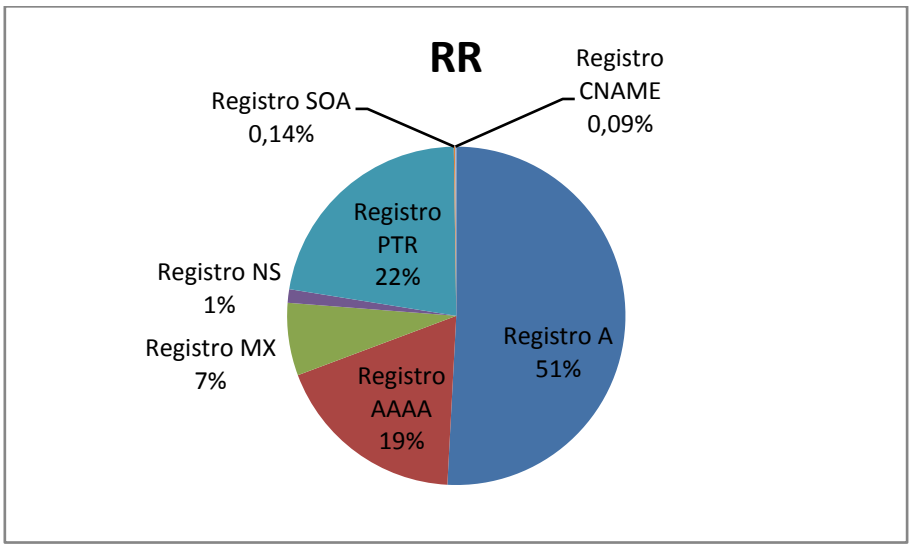
En el Ejemplo 8-5 se muestra una forma básica de realizar una firma de zona. Los parámetros especificados en el ejemplo son: el directorio donde se encuentran las claves generadas, como las del Ejemplo 8-4, el origen −ápex− de la zona y el archivo conteniendo los registros de recursos. El programa de firmado devolverá como salida un archivo nuevo de zona con nombre *zonefile.signed*. Una observación para hacer acerca de este archivo firmado, es que los nombres de dominio aparecen ordenados según el orden canónico, lo cual es imprescindible para la generación de los registros NSEC. En caso de estar utilizando NSEC3, el espacio de nombres hash definido en la sección 6.3.1 - también resultará ordenado canónicamente junto con el espacio de nombres de dominio. En caso de que la zona pertenezca a una cadena de confianza, se deben incluir los registros DS correspondientes a las zonas hijas. La forma más fácil de lograr esto es simplemente insertar los RR DS de las zonas hijas en el archivo de zona sin firmar, y al ejecutar dnssec-signzone estos RR se firmarán como cualquier otro registro. Del mismo modo, el archivo con los RR DS generados junto con el archivo firmado, debe ser enviado a la zona padre para que este los agregue a su zona y le adjunte sus firmas digitales.

8.2 - Replica de un servidor de SeCIU

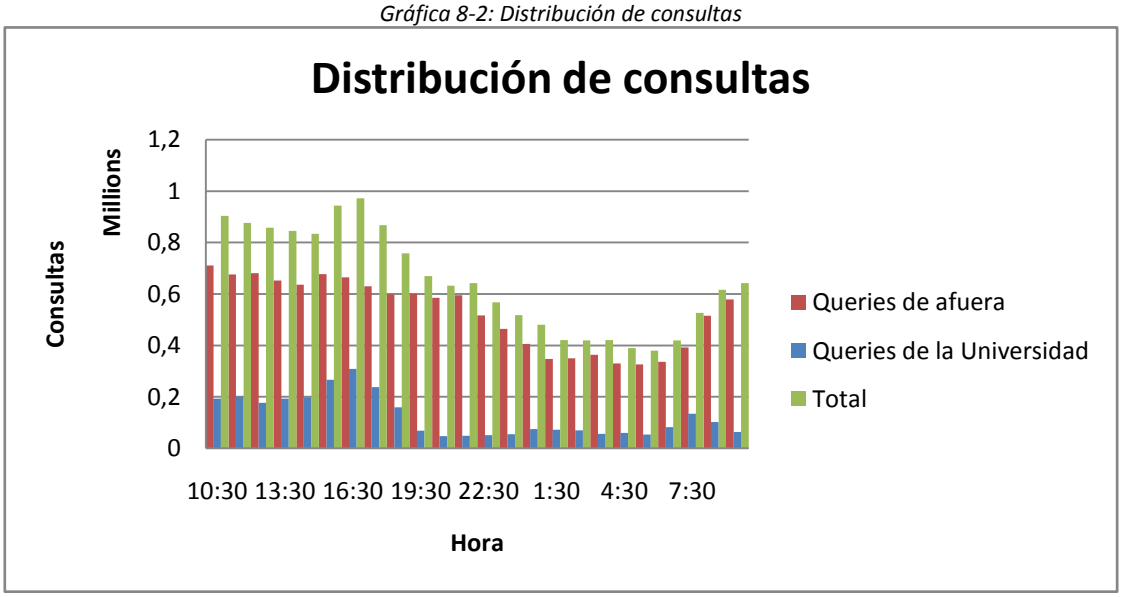
Si bien esta prueba se realizó luego de haber terminado el resto de las experiencias que describiremos más adelante, la presentaremos primero para introducir resultados estadísticos de un servidor en producción. En particular, el servidor que SeCIU utiliza para atender, entre otras zonas, a la uy, edu.uy, org.uy, mil.uy, net.uy, gub.uy y com.uy y varios dominios reversos bajo in-addr.arpa. De todas estas, SeCIU administra autoritativamente todas menos la com.uy, para la cual delegó su autoridad a ANTEL. Este servidor tiene permitido ofrecer servicios de búsqueda recursiva a las consultas que provengan de la Red Académica Uruguaya −RAU−, la cual tiene asignado el bloque IPv4 164.73.0.0/16. Las consultas provenientes de direcciones no pertenecientes a la RAU correspondientes a dominios para los cuales SeCIU no es autoritativo serán respondidas indicando con una flag que la recursión no esta disponible −*Recursion not Available*− y una referencia al siguiente servidor de nombre en caso de que no pueda responder autoritativamente.

Como ya se mencionó, el personal de SeCIU nos brindó trazas capturadas sobre este servidor, estas contienen todo el trafico DNS sobre dicho servidor en el correr de un día entero, capturando tanto las consultas entrantes, las respuestas y las consultas iterativas realizadas por el servidor. Con esas trazas se realizaron estadísticas acerca de qué tipo de consultas son realizadas y de donde provienen, y qué registros son solicitados. Se observa en la Gráfica 8-1 que la mitad de los registros solicitados son del tipo A, seguido del RR PTR asociado a la resolución reversa de direcciones, y el registro AAAA el cual irá ganando terreno a medida que el despliegue de IPv6 siga avanzando.

Gráfica 8-1: Distribución de RR

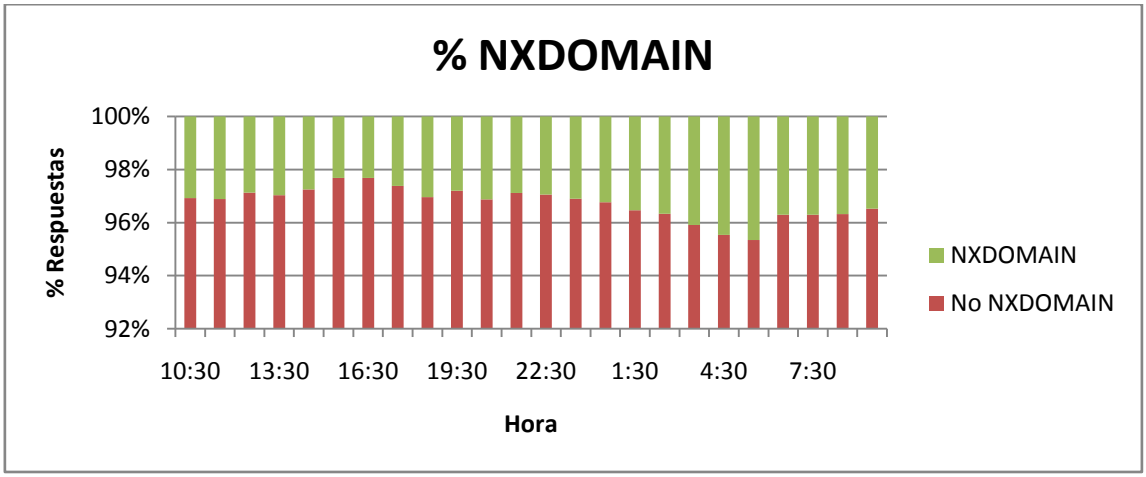


Gráfica 8-2: Distribución de consultas



En la Gráfica 8-2 tenemos la distribución de consultas realizadas a SeCIU en el correr de un día. Se diferenciaron las consultas realizadas desde las direcciones de red asignadas a la RAU y las realizadas desde otras direcciones. Si bien las consultas de la universidad representan el 20% del total, estas son las que más recursos consumen dado que se resuelven de forma recursiva.

Gráfica 8-3: % NXDOMAIN



Para replicar de forma más exacta el comportamiento del servidor de SeCIU, se debió considerar las consultas que generan una respuesta NXDOMAIN. Esto es principalmente importante cuando se utiliza NSEC3 para la autenticar respuestas de no existencia. Este porcentaje de consultas trae asociado un incremento en el consumo de CPU, debido a las operaciones criptográficas que deben realizarse durante la validación de los registros NSEC3 –ver sección 6.3.4 - –, que impactará de forma negativa en la performance del servidor. Considerando los resultados de la Gráfica 8-3,

las trazas utilizadas en las pruebas de recursividad se generaron conteniendo un 2% de consultas de registros no existentes en la zona.

8.2.1 - Maqueta SeCIU

A partir de los datos obtenidos de las trazas brindadas por SeCIU, se realizó la prueba de simular el servidor de SeCIU con nuestros equipos, para determinar si nuestra computadora UbuntuDesktop está en condiciones de ofrecer los servicios de búsqueda recursiva que el servidor de SeCIU ofrece para la RAU. Para esto, se filtró utilizando el Wireshark una sub-traza conteniendo todas las consultas entrantes provenientes de la red universitaria:

```
ip.src==164.73.0.0/16 && ip.dst==Server_SeCIU && dns.flags.response==0
```

Utilizando el programa tcpreplay, se modificaron las direcciones IP y MAC de esta nueva traza para poder ser utilizada en nuestros equipos.

Se utilizó la computadora UbuntuDesktop −164.73.45.145− para simular el servidor de SeCIU, mientras que se dispuso a la UbuntuServer −164.73.45.146− para que realizara las consultas sobre la primera.

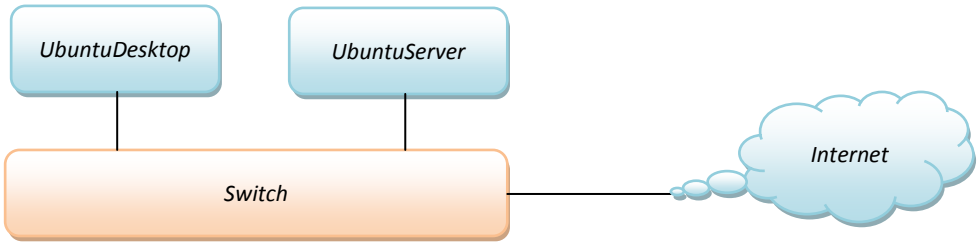


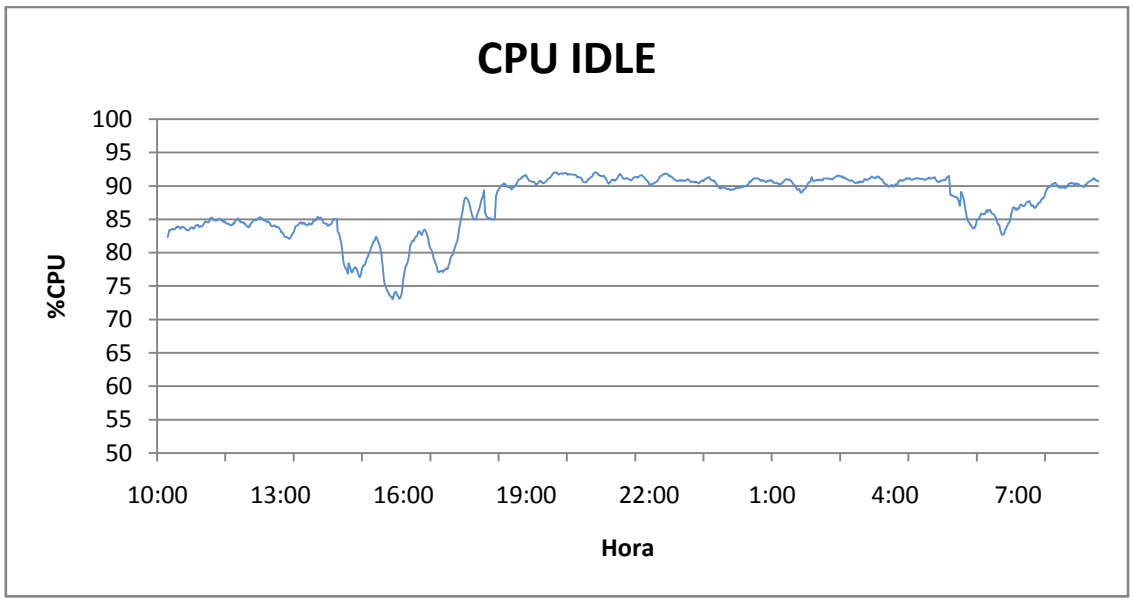
Figura 8.1: Maqueta replica SeCIU

Gráfica 8-4: Maqueta SeCIU Consultas

En la Gráfica 8-4 se representan las consultas por segundo −qps− realizadas a nuestro servidor en función de la hora. La cual es consistente con la Gráfica 8-2, que muestra las consultas realizadas desde la RAU.

Cabe aclarar que esta experiencia se realizó utilizando únicamente las consultas provenientes de la RAU, que se resuelven recursivamente, descartando las originadas fuera de dicha red. Como veremos más adelante, estas consultas son las que más recursos consumirán y determinarán la performance del servidor. Utilizaremos los resultados obtenidos en esta experiencia como base para comparar servidores recursivos operando DNSSec.

Gráfica 8-5: Maqueta SeCIU CPU IDLE



Se midió el consumo de CPU durante el tiempo que se realizó la medida, encontrando que las caídas de tiempo de CPU idle corresponden con picos de consultas. Como veremos en las pruebas sobre el servidor recursivo, este es el que más recursos de CPU y ancho de banda consumirá. La importancia de esta medida es que nos permite asegurar que nuestro equipo es capaz de soportar el nivel de carga de SeCIU.

8.3 - Servidor de nombre autoritativo

Esta prueba consistió en recrear un servidor autoritativo para medir su performance ante diferentes niveles de tráfico. La computadora UbuntuDesktop cumplió el rol de servidor autoritativo, mientras que utilizamos al equipo Cliente para cargar al servidor con las trazas de prueba. Se configuró al servidor para que funcionara como autoritativo para las zonas de SeCIU que teníamos disponibles, estas

eran las zonas uy, edu.uy, org.uy, net.uy, mil.uy y gub.uy. Las trazas utilizadas para esta prueba tuvieron que ser generadas por nosotros, ya que era necesario tener dos juegos de consultas de iguales características, siendo la única diferencia entre ellos que uno solicitaba registros DNSSec y el otro no. Esto nos permitió asegurar que las diferencias entre la performance del equipo sirviendo zonas firmadas y no firmadas se debiera únicamente a los registros extra que circulaban.

Como se vio en la prueba anterior, existe un 2% de consultas que generan respuestas de no existencia. Por lo que las consultas utilizadas debían contemplar este porcentaje, ya que estas generarían respuestas conteniendo registros NSEC y sus firmas. Este porcentaje será de mayor importancia al utilizar NSEC3 en la autenticación de no existencia, lo que veremos posteriormente en otra prueba.

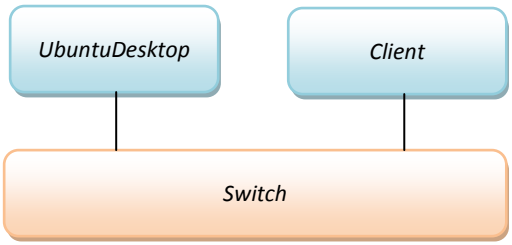


Figura 8.2: Maqueta Autoritativo

En la Figura 8.2 se representa la maqueta utilizada para esta prueba. La máquina Cliente simuló el mundo exterior realizando consultas a nuestro servidor autoritativo. También, se utilizó la maquina Cliente para capturar todo el tráfico, de modo de no afectar la performance del servidor autoritativo. Una observación a tener en cuenta es que dado que las consultas son replicadas con tcpreplay con la IP origen de Cliente, este equipo desconocerá las respuestas que UbuntuDesktop le envíe, devolviendo paquetes ICMP anunciando que no se está escuchando el puerto destino. Para evitar que estos paquetes contaminen las medidas, se debe bloquear la salida de ICMP de la computadora Cliente:

```
iptables -A OUTPUT -o eth0 -p icmp -j  DROP
```

La configuración en BIND del servidor de nombre autoritativo se puede ver en el Anexo F.

8.3.1 - Medidas Servidor Autoritativo

En primer lugar se realizaron las pruebas con las zonas sin firmar de modo de obtener un punto de comparación de la performance de los servidores. Se tomaron 30 medidas desde 200 pps hasta 10kpps, registrando la cantidad de consultas enviadas, consultas contestadas, Mbps cursados, porcentaje de CPU idle y porcentaje del CPU utilizado por el servidor de nombre named. El tiempo de medida fue fijado cambiando la cantidad de paquetes que se envían en forma proporcional a la cantidad de

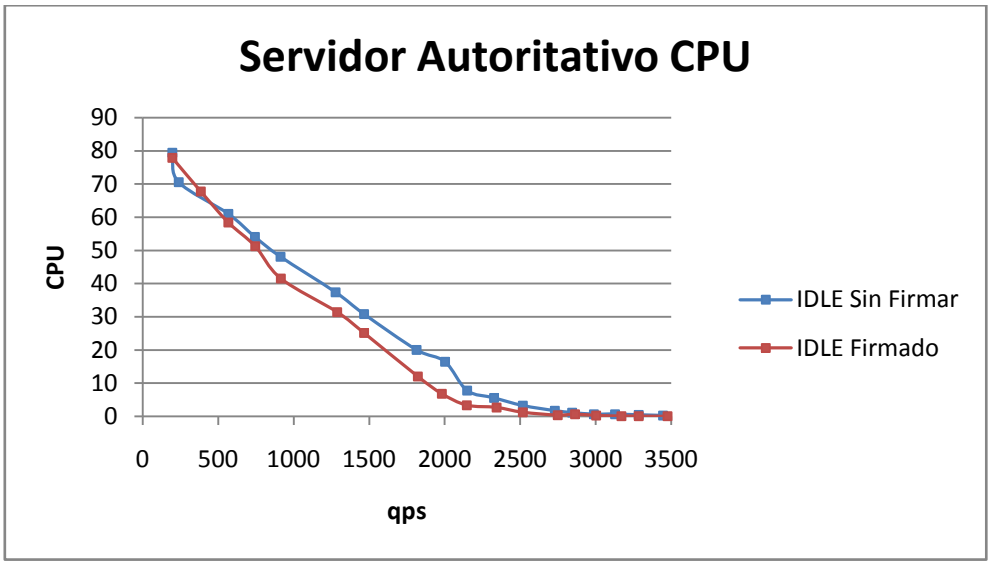
paquetes por segundo, logrando siempre una duración de las medidas de 40 seg. Las medidas de CPU se hicieron sobre este intervalo de tiempo de la siguiente manera:

```
sar 1 35
pidstat -p PID_NAMED 1 35
```

Ambos programas corren independientemente y devuelven cada uno la media sobre 35 medidas tomadas de a 1 segundo. Si bien la propia ejecución de estas aplicaciones repercute sobre la performance del equipo, se utilizó siempre este mismo esquema de modo de obtener resultados comparables.

Una vez terminadas las pruebas con las zonas sin firmar, se reconfiguró el servidor de nombre para sirva las zonas firmadas según se describe en el Anexo F. Realizadas todas estas pruebas se obtuvieron los siguientes resultados.

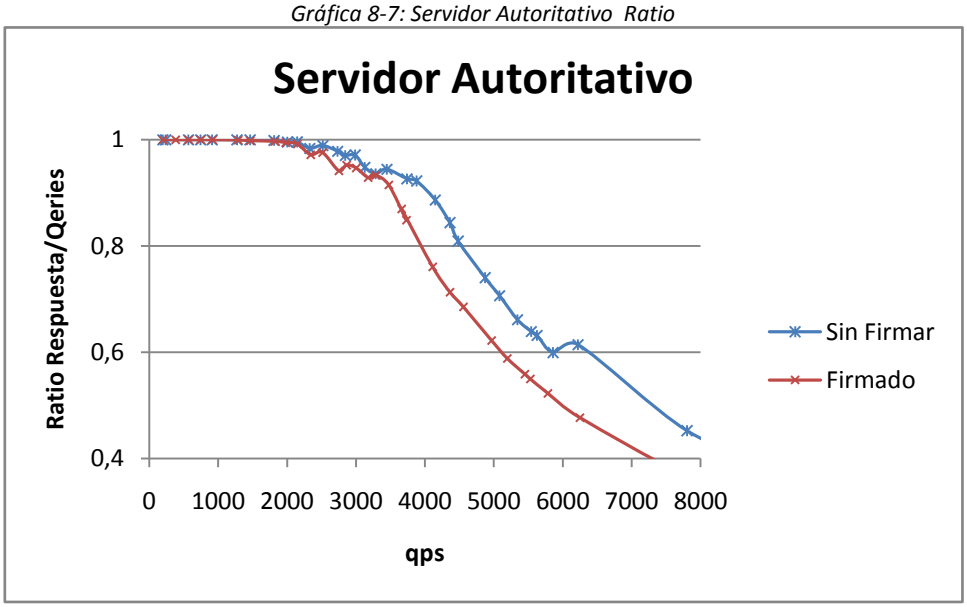
Gráfica 8-6: Servidor Autoritativo CPU



Se observa en la Gráfica 8-6 que la curva de porcentaje idle firmado muestra un leve decaimiento frente a su contraparte sin firmar. Este leve decaimiento puede deberse a que se deben enviar más registros en cada respuesta, impactando en el trabajo que debe realizar el servidor.

Calculamos la relación entre la cantidad de respuestas y la cantidad de consultas –*ratio*– para medir la performance del equipo UbuntuDesktop frente al incremento de paquetes por segundo, observar que en UDP un paquete corresponde a una consulta, por lo que los nombraremos indistintamente.

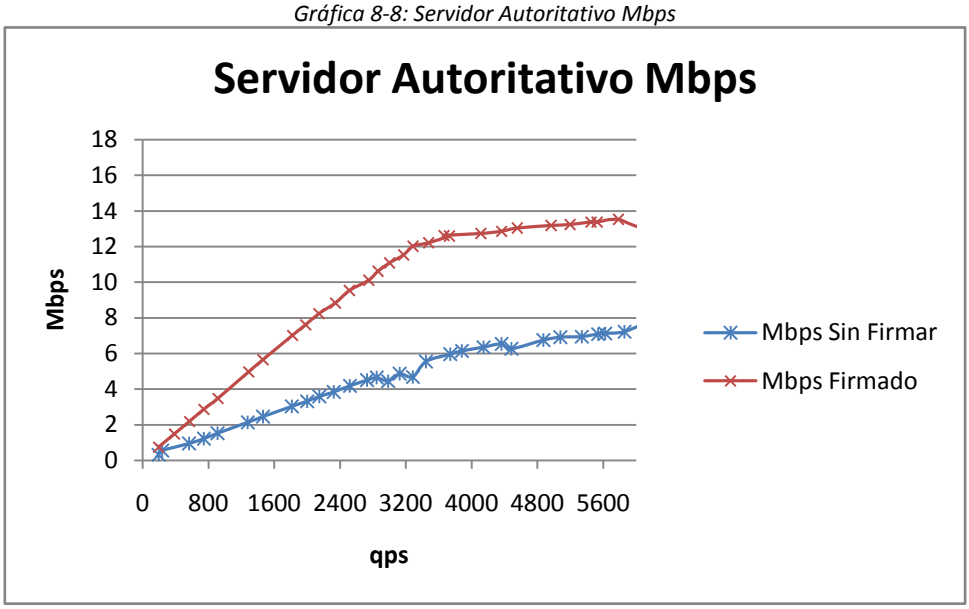
Gráfica 8-7: Servidor Autoritativo Ratio



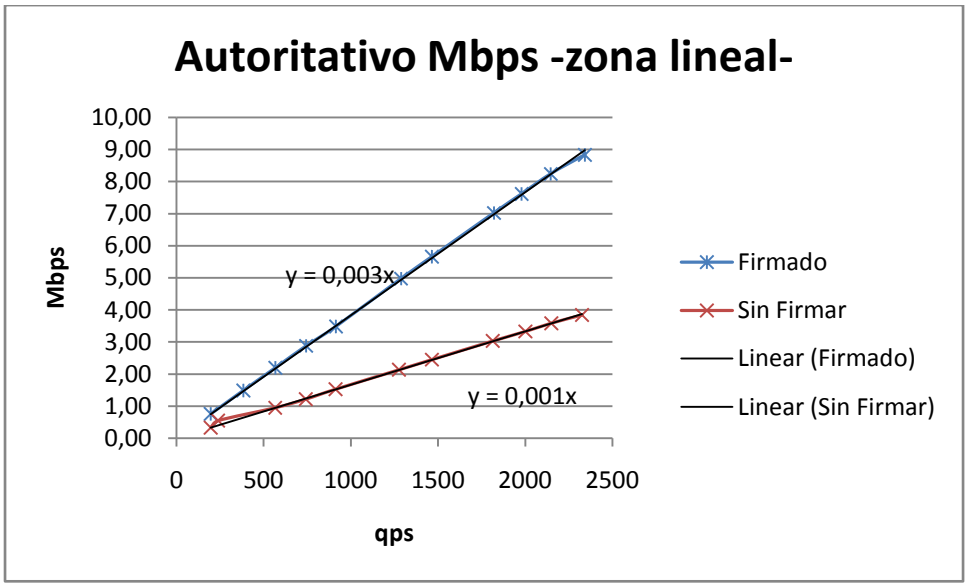
En la Gráfica 8-7 se representa el *ratio* calculado antes y después de firmar. En esta se muestra que a partir de los 2000 qps ya se comienza a percibir perdidas de paquetes, y que a los 3000 qps la caída comienza a pronunciarse, momento en el cual se puede decir que el porcentaje de CPU idle ya está en cero, lo que indica que en nuestro equipo UbuntuDesktop la performance está limitada por la CPU la cual actúa como cuello de botella.

Para medir como incrementa el ancho de banda consumido debido a las firmas, veamos la Gráfica 8-8. En el primer segmento de curva −que llamaremos zona lineal− aún tenemos tiempo de CPU disponible, por lo que era de esperar la relación lineal que se observa en la gráfica entre las qps y los Mbps. Comparando las pendientes de estas curvas, observamos que operar una zona firmada incrementa el ancho de banda consumido. En el caso del servidor autoritativo, esto se debe a los registros firmados que se adjuntan a las respuestas. En la segunda zona de esta gráfica, la zona de saturación, podemos observar que las curvas tienden a ser paralelas. La razón de esto reside en el hecho de que a partir del punto de saturación −3200 pps− la cantidad de respuestas que el servidor es capaz de enviar es constante. Esto determina que la pendiente de la zona de saturación de la Gráfica 8-8 este dada por el aumento de qps y no por las respuestas −ya sean firmadas o no−.

Gráfica 8-8: Servidor Autoritativo Mbps



Gráfica 8-9: Autoritativo Mbps Zona lineal



En la Gráfica 8-9 se realizó un análisis de los puntos de la zona lineal de esta prueba. Encontramos que la relación entre la pendiente de las curvas −Mb/query− de los datos firmados contra los no firmados es aproximadamente el doble: 0,0038/0,0017≈2, dato que utilizaremos en comparaciones con otras pruebas.

8.4 - Servidor recursivo

Si bien los servidores autoritativos representan una parte importante en el DNS, los que hacen la mayoría del trabajo en nombre de los clientes son los servidores

recursivos. Los servidores que proveen los proveedores de Internet −ISPs−, resuelven todas las consultas de sus clientes. Algunas empresas también disponen de servidores recursivos públicos, como por ejemplo los servidores DNS de Google, los cuales atienden 70 mil millones de consultas diarias [35].

Con el objetivo de acercarnos al comportamiento y a las características de tráfico de estos servidores, propusimos una serie de medidas variando diferentes parámetros. En primer lugar se configuró nuestro servidor UbuntuDesktop para que no responda ninguna pregunta desde su cache. Esto se hizo fijando los TTL de nuestros registros en 0, obligando al servidor recursivo a resolver todas las consultas que recibe, aunque sean repetidas. Trabajos relacionados [36] indican que un servidor recursivo responde más del 70% de sus consultas desde cache −en promedio y luego de estar un tiempo en operación−, por lo que nuestra segunda prueba con estos servidores consistió en fijar un porcentaje fijo de respuestas que provenientes del cache.

Otro parámetro que también debió considerarse fue el delay, si bien el delay que existe entre los equipos de la universidad y el servidor recursivo de SeCIU es del entorno de los 10ms, el delay hacia los servidores autoritativos u otras zonas lejanas geográficamente pueden superar ampliamente los 100ms. Para estudiar el efecto que este parámetro puede tener sobre un servidor recursivo, consideramos sobre un punto de operación −pps− del servidor con cache, diferentes delays para medir el efecto que este puede tener sobre la performance.

La maqueta utilizada en las pruebas de recursividad se representa en la Figura 8.3. Ahí se muestran las computadoras utilizadas y los roles que cumplieron durante estas pruebas.

8.4.1 - Servidor recursivo sin cache

Para la realización de estas medidas, se dispuso una maqueta que contenía además del servidor recursivo, servidores autoritativos de las zonas raíz, uy y edu.uy, de manera de obtener al menos 3 iteraciones en la resolución de los nombres de dominio. En el último nivel de nuestro árbol DNS, se configuró una computadora para servir las zonas fing.edu.uy. y primaria.edu.uy. Estas zonas se crearon a partir de un programa en C, el cual genera dominios dentro de estas zonas. El servidor UbuntuDesktop se configuró para actuar de forma recursiva A partir de estas nuevas zonas, al igual que se hizo con la prueba del autoritativo, se generó una traza que contenía consultas aleatorias por registros de dominios de estos últimos niveles. Del mismo modo, replicamos el tráfico con tcpreplay obteniendo los siguientes resultados.

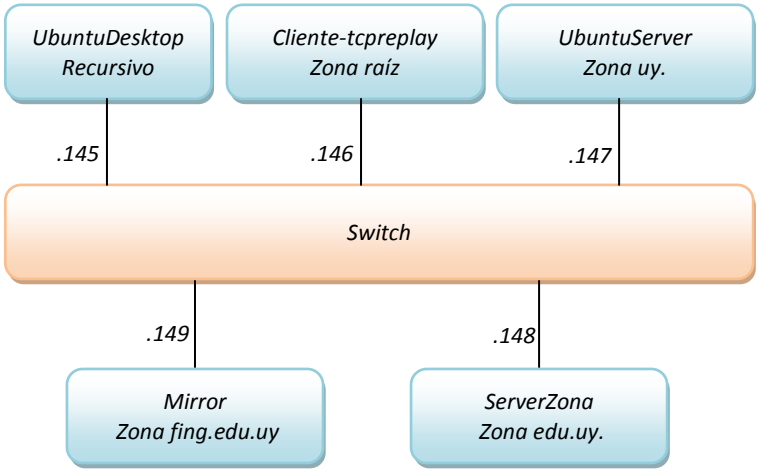
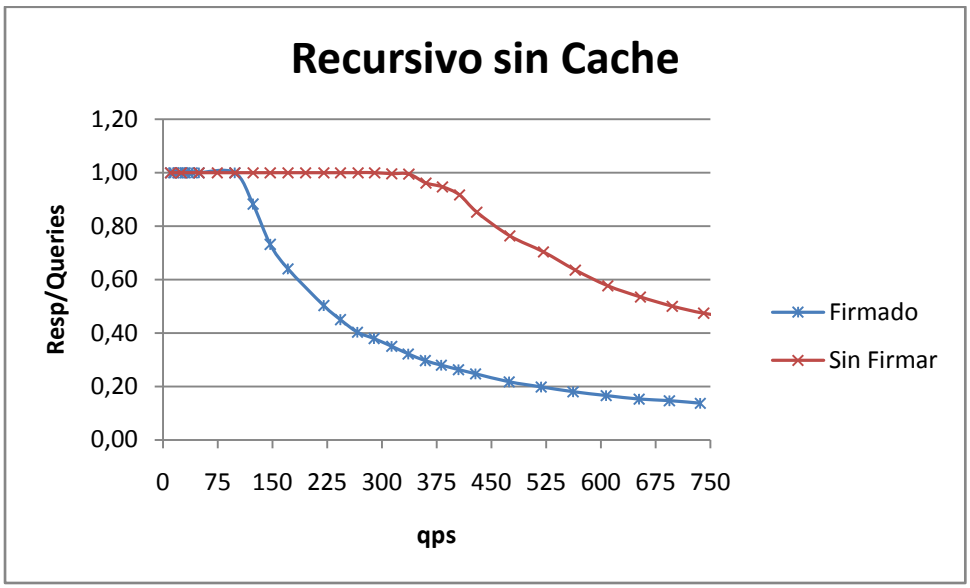


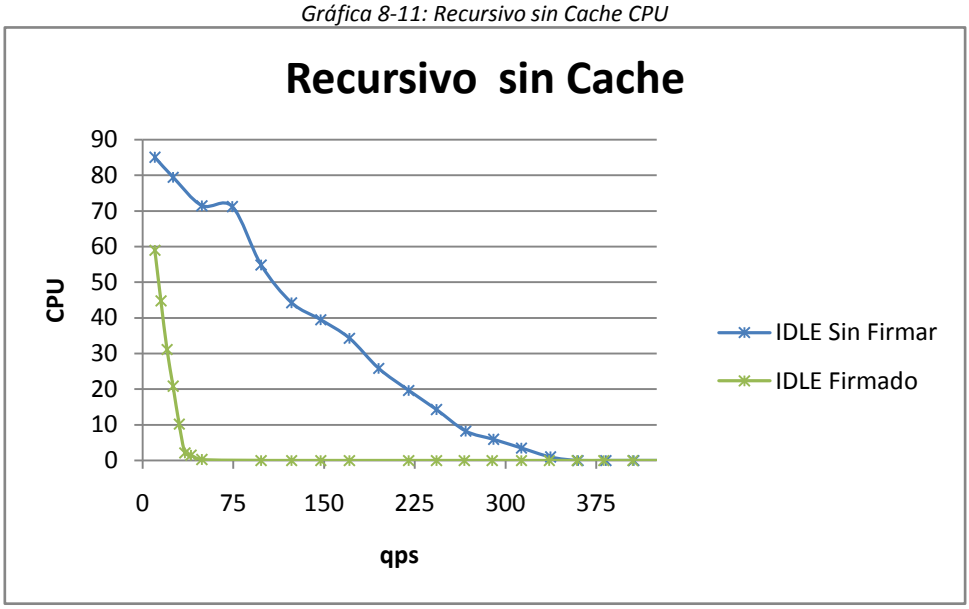
Figura 8.3: Maqueta Recursivo

Gráfica 8-10: Recursivo sin Cache Ratio



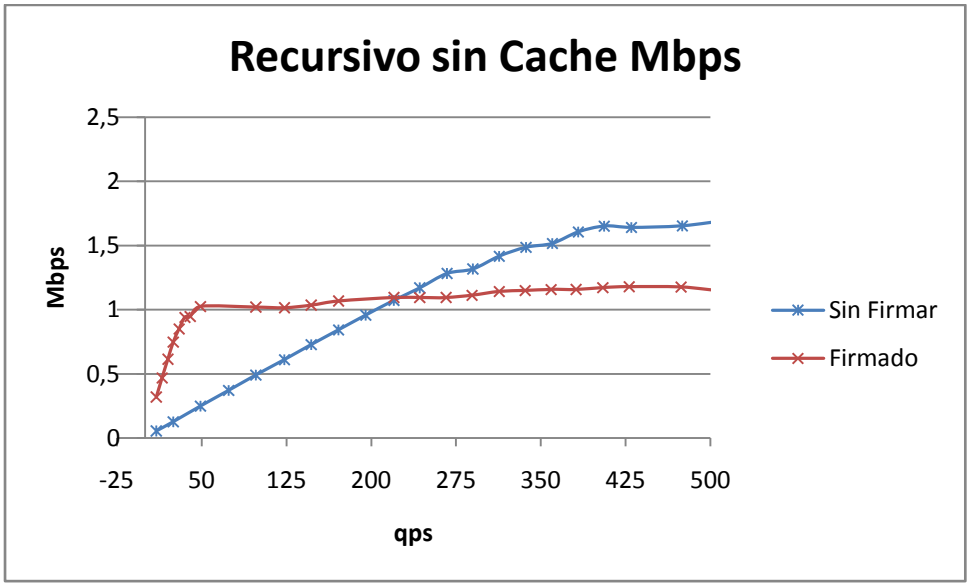
Al igual que en la prueba del servidor autoritativo, se realizaron medidas de performance para distintos volúmenes de consultas por segundo. Comparando la curva sin firmar de esta experiencia y su correspondiente en la prueba del servidor autoritativo −Gráfica 8-7−, se observa como la curva del servidor recursivo se deteriora mucho antes que la del servidor autoritativo. Esto es un claro indicio de que un servidor recursivo consume muchos más recursos. Comparando las curvas firmadas y sin firmar de la Gráfica 8-10, se observa una degradación más pronunciada de la primera en comparación con la curva de la segunda.

Gráfica 8-11: Recursivo sin Cache CPU



Nuevamente, podemos concluir que la diferencia apreciada en la Gráfica 8-10 se debe al consumo de CPU, el cual en 100% de recursividad −TTL=0− se incrementa de forma alarmante. La marcada separación entre los consumos de CPU firmados y no firmados se debe a la verificación de las firmas criptográficas que el servidor recursivo debe realizar.

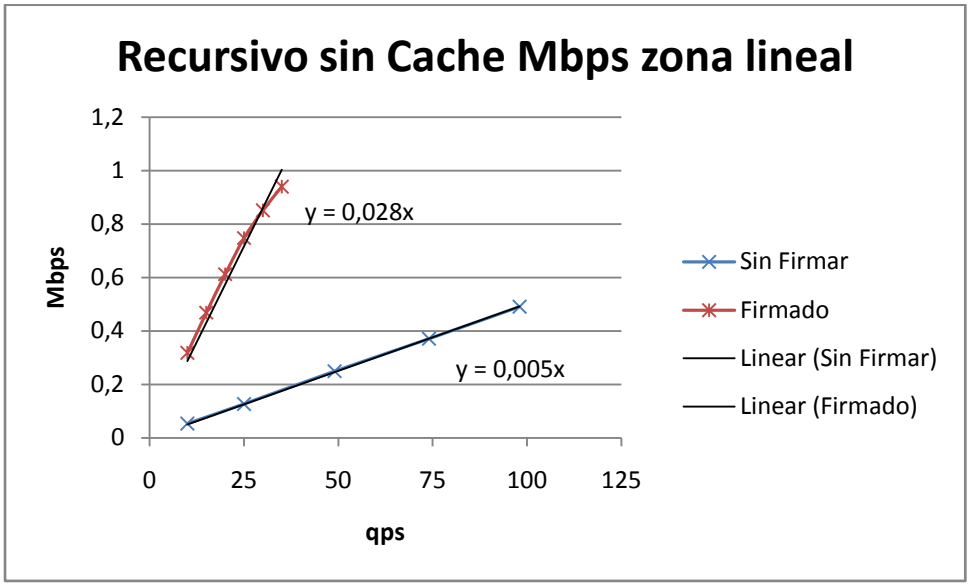
Gráfica 8-12: Recursivo sin Cache Mbps



En términos de ancho de banda encontramos que, como era de esperar, el ancho de banda requerido por cada consulta se incrementó. También se observó que la prueba con zonas firmadas saturó a una velocidad de 50 qps, nuevamente debido a la falta de capacidad de procesamiento del CPU en el servidor recursivo. Nuevamente podemos observar aquí el efecto de saturación observado en la prueba del servidor

autoritativo, con la clara diferencia de que la curva firmada presenta una seria degradación.

Gráfica 8-13: Recursivo sin Cache Mbps Zona Lineal



Para comparar el incremento en ancho de banda entre firmado y no firmado, analicemos la zona lineal que se muestra en la Gráfica 8-13. La relación entre las pendientes de las dos curvas es 0,0286/0,005=5,72 representa el aumento del consumo de ancho de banda promedio por paquete. Este incremento esta dado no solo por las firmas digitales de los registros solicitados por el cliente, sino también por los registros necesarios en la construcción de la cadena de confianza.

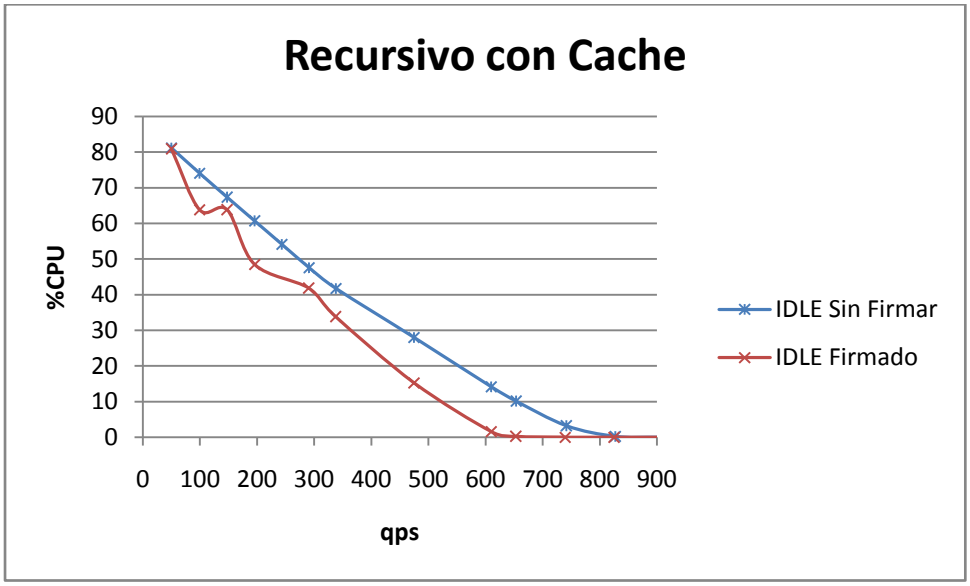
8.4.2 - Servidor Recursivo con Cache

La segunda prueba que se realizó sobre esta misma maqueta, Figura 8.3, es la de recursividad con cache. Es de esperar que cualquier servidor recursivo guarde en cache los registros que este resuelve, lo que trae asociado un incremento en la eficiencia del mismo. Para establecer el nivel de cache −porcentaje de respuestas que se contestan desde el cache−, nos referimos a trabajos anteriores [36], que marcan porcentajes mayores al 70%, valor que consideramos razonable y fue utilizado en esta experiencia.

Para la realización de esta prueba, encontramos que era necesario primeramente “llenar” el cache con el 70% de las consultas que se iban a realizar. Para esto utilizamos un programa en C que genera tráfico, seleccionando de forma aleatoria el 70% de las consultas totales. Este tráfico era primeramente replicado −utilizando tcpreplay− sobre el servidor recursivo, de modo que las respuestas resueltas fueran almacenadas en cache. Esto requeriría que los TTL de los recursos fueran mayores a 0.

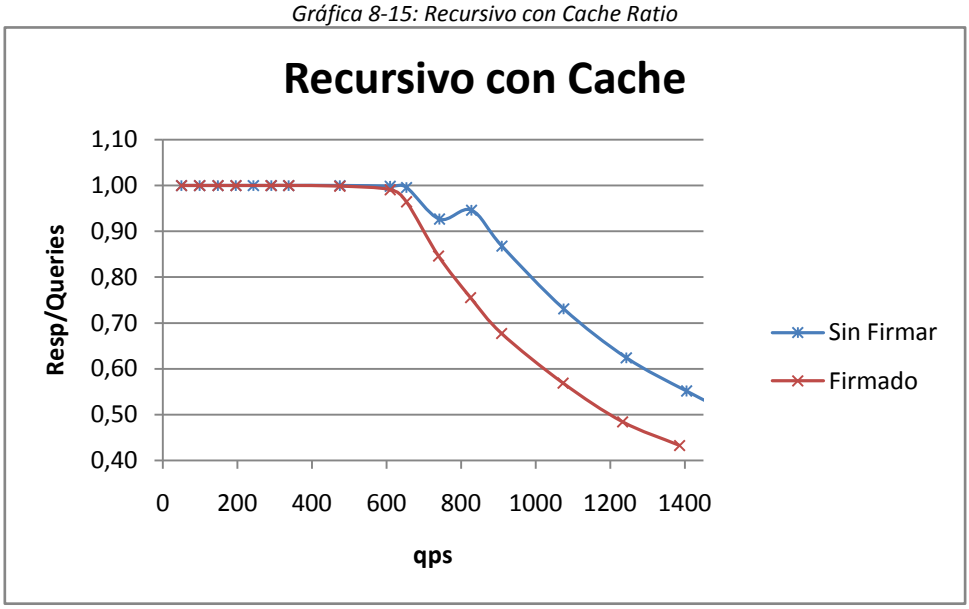
Se generó la zona nuevamente con TTL de 2 horas, lo que daba tiempo más que suficiente para llenar el cache y realizar la medida. Este procedimiento se tuvo que repetir para cada medida, ya que en la medida siguiente las consultas realizadas anteriormente ya estaban todas en cache. Para vaciar el cache y comenzar devuelta con el llenado del 70% se utilizó el comando *rndc flush,* ver Anexo F, el cual vacía el cache del BIND.

Gráfica 8-14: Recursivo con Cache CPU



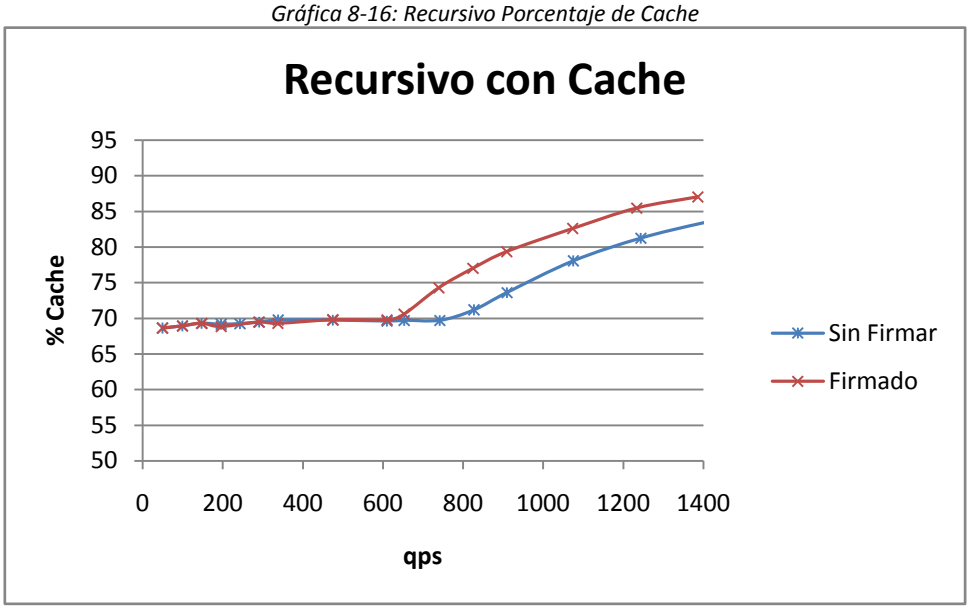
Este 70% de respuestas que ya están almacenadas en el cache, pueden ser devueltas al cliente sin necesidad de ningún procesamiento extra, lo que permite que el servidor pueda responder todas las consultas hasta un número de pps mucho mayor. Comparando la Gráfica 8-14 con la Gráfica 8-11, veremos que sin cache a los 75 pps el CPU ya se encuentra saturada, mientras que con cache se puede llegar hasta los 600 pps sin saturar el CPU. En términos numéricos, el servidor recursivo con cache es 8 veces más eficiente que sin cache.

Gráfica 8-15: Recursivo con Cache Ratio



Observando ahora la Gráfica 8-15, vemos como el punto donde comienzan las perdidas es casi el doble que sin cache, en el caso no firmado, y 7.5 veces mayor en el caso firmado. Esta diferencia tan pronunciada en el caso firmado marca la carga que impone la implementación de DNSSec en los servidores recursivos y como se ve afectado el rendimiento a medida que varía el cache –en nuestro estudio 0 y 70%- Habiendo visto esto, resulta de interés ver cómo se comporta el cache a medida que se aumentan las qps y comienzan las pérdidas.

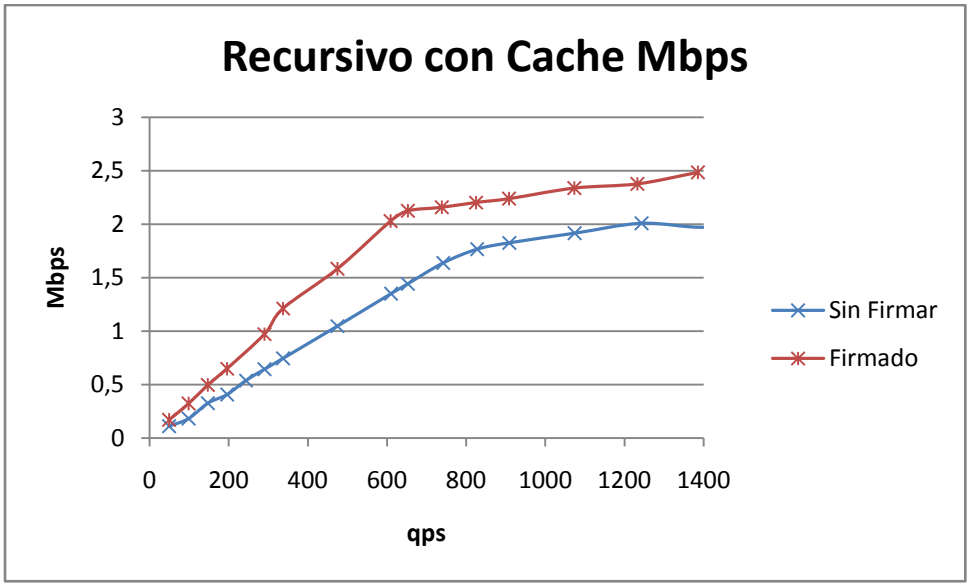
Gráfica 8-16: Recursivo Porcentaje de Cache



En la Gráfica 8-16 observamos como varia la cantidad de respuestas que son devueltas desde cache. Mientras no haya pérdidas, las respuestas de cache se mantienen en el entorno del 70%. Pero al comenzar el momento en el cual el servidor no logra responder todas las consultas, nos encontramos con que el servidor responde

cada vez más consultas de cache. Si bien la proporción de consultas contestadas es cada vez menor, las pocas que se responden se originan más del cache que de una resolución recursiva.

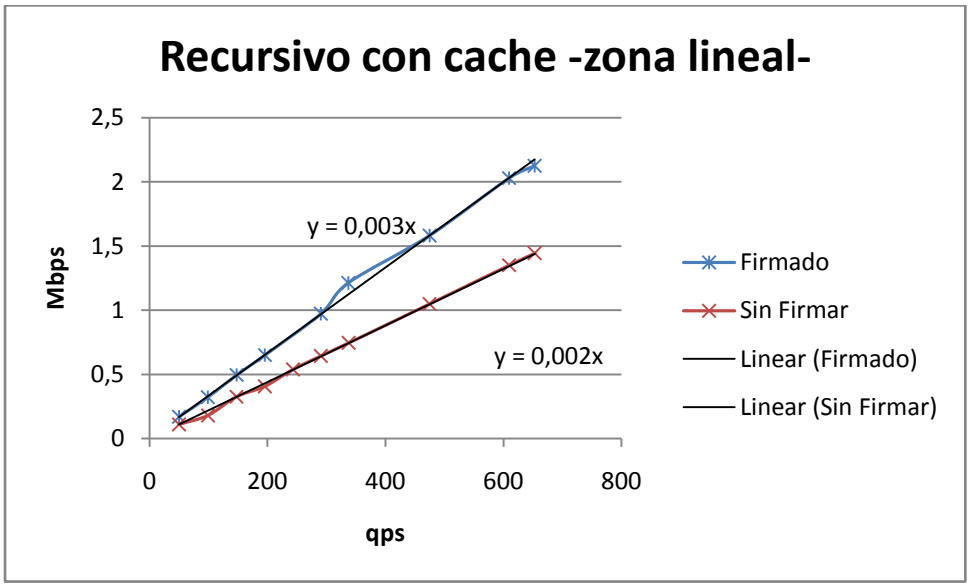
Gráfica 8-17: Recursivo con Cache Mbps



Ahora observaremos como varía el ancho de banda a medida que aumentamos las qps. Para esto nos referiremos a la Gráfica 8-17.

Para finalizar, al igual que en los casos anteriores observaremos si es posible aproximar la zona sin pérdidas por una recta.

Gráfica 8-18: Recursivo con Cache Mbps Zona lineal



Como podemos observar en la Gráfica 8-18, los puntos se pueden aproximar de buena manera por una función lineal. Este comportamiento puede probar ser de utilidad a la hora del diseño del servidor y la red, ya que sería posible estimar el ancho de banda necesario sabiendo las características del servidor y de esta manera dimensionar la red.

8.4.3 - Servidor Recursivo con Delay

Como se describió al comienzo, esta prueba se realizó con el fin de estudiar el comportamiento del servidor recursivo sin firmar y firmado cuando existen demoras en la red. Para el servidor sin firmar tomamos dos puntos de operación a 500 pps y a 900 pps. Se tomaron estos puntos en particular ya que a 500 pps el servidor se encuentra en un punto normal de funcionamiento –sin pérdidas– y con un 70% de respuestas de cache, y a 900pps es el punto donde nos encontramos en un 5% de pérdidas aproximadamente.

Se siguió la práctica realizada en la prueba del servidor recursivo con cache, lo que implicó: un TTL de dos horas para todas la zonas y que en cada medida se borraba y llenaba cache a un 70%, teniendo el cuidado de llenar cache a una velocidad suficientemente baja para no tener pérdidas. En el servidor autoritativo utilizamos las funcionalidades de calidad de servicio –QoS– de Linux, en particular la política "netem" del paquete tc –traffic control–, para introducir un retardo controlado en las respuestas del mismo. Finalmente el comando utilizado para agregar este delay fue:

```
tc qdisc add dev eth0 root netem delay DELAY_ms
```

Gráfica 8-19: CPU

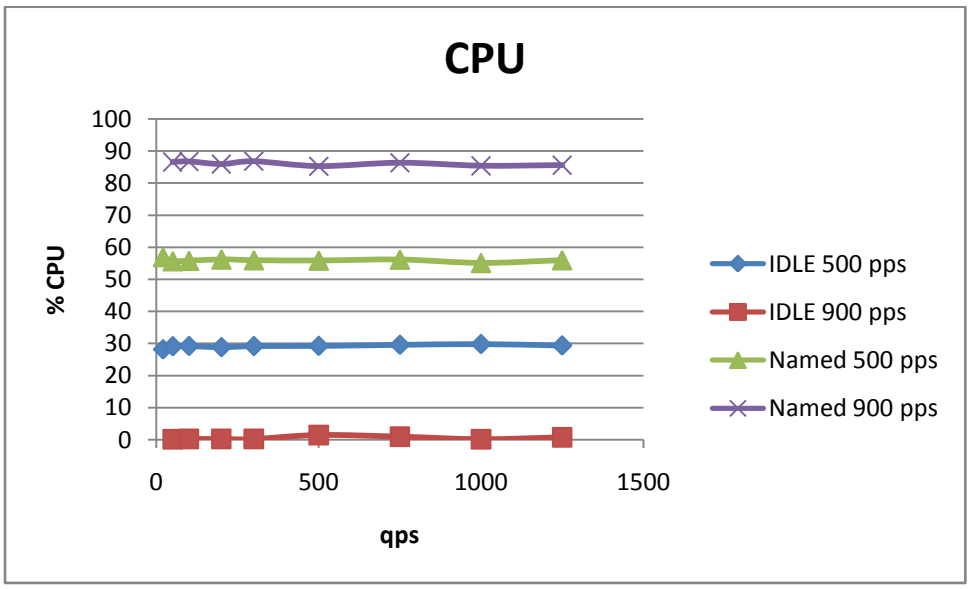


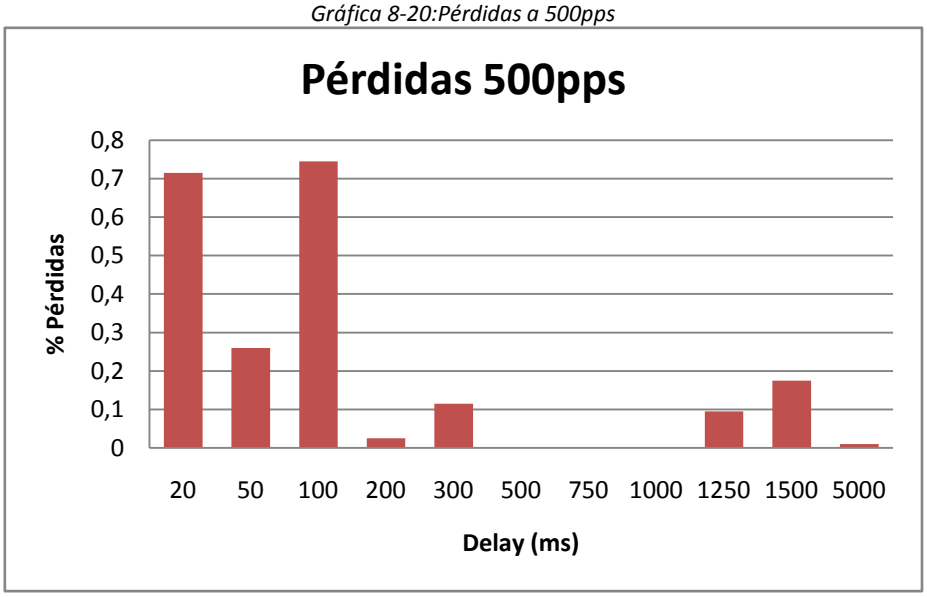
Tabla 8-1: Referencia

	%Pérdidas	IDLE	Named
Sin Delay	0	28	56.7
Delay 20ms	0.7	28,11	56.96
Delay 500..1000	0	29.2	56.15
Delay 1500	0.17	29.92	55.94

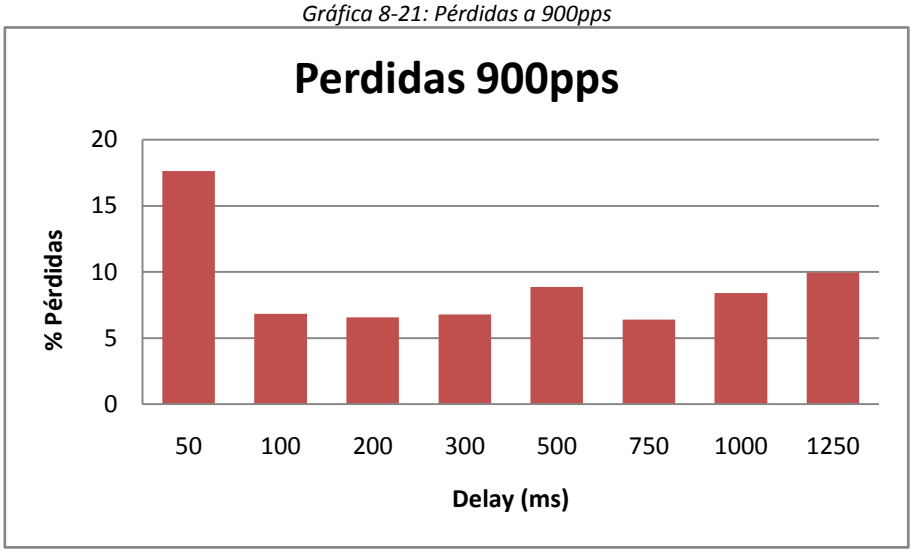
Tabla 8-2: Referencia

	Pérdidas %	IDLE	Named
Sin Delay	0.05	0.24	84.61
Delay 50ms	17.62	0.12	86.55
Delay 1250	6.55	0.2	85.88

Gráfica 8-20:Pérdidas a 500pps

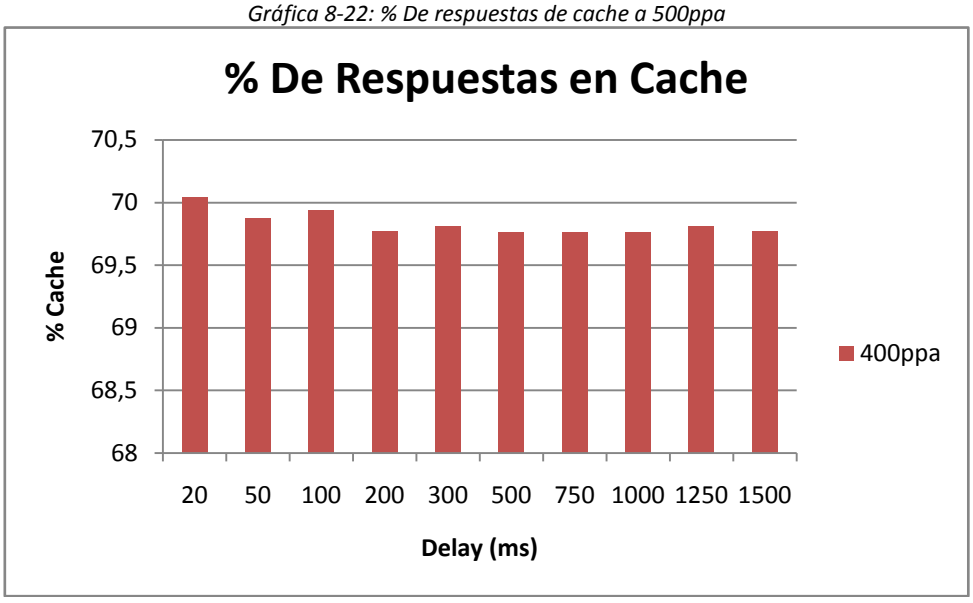


Gráfica 8-21: Pérdidas a 900pps

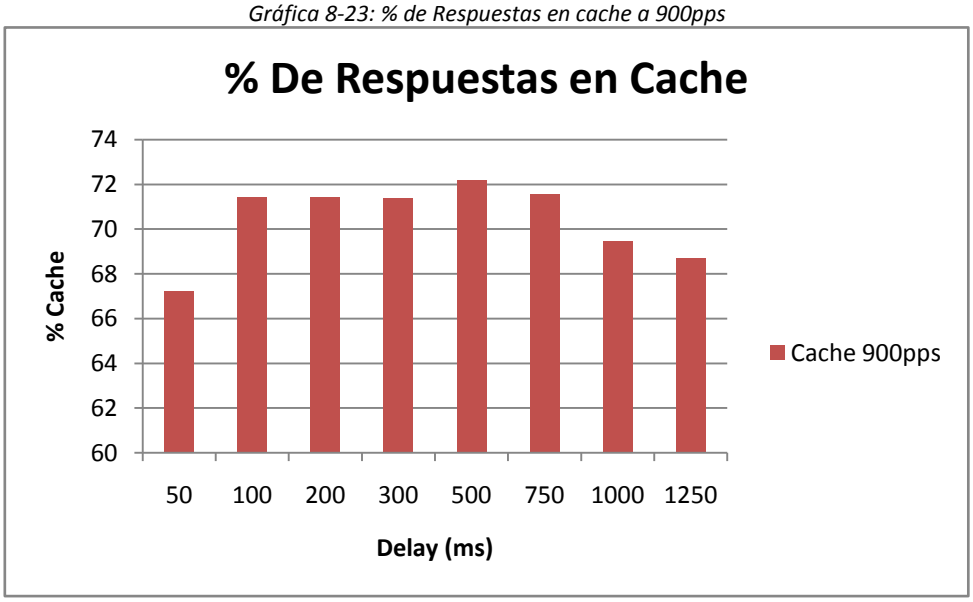


En base a lo observado en las gráficas no se visualizan grandes cambios en el consumo de CPU en el servidor recursivo, así como tampoco se observa una variación relevante en las pérdidas al aumentar el delay.

Gráfica 8-22: % De respuestas de cache a 500ppa



Gráfica 8-23: % de Respuestas en cache a 900pps



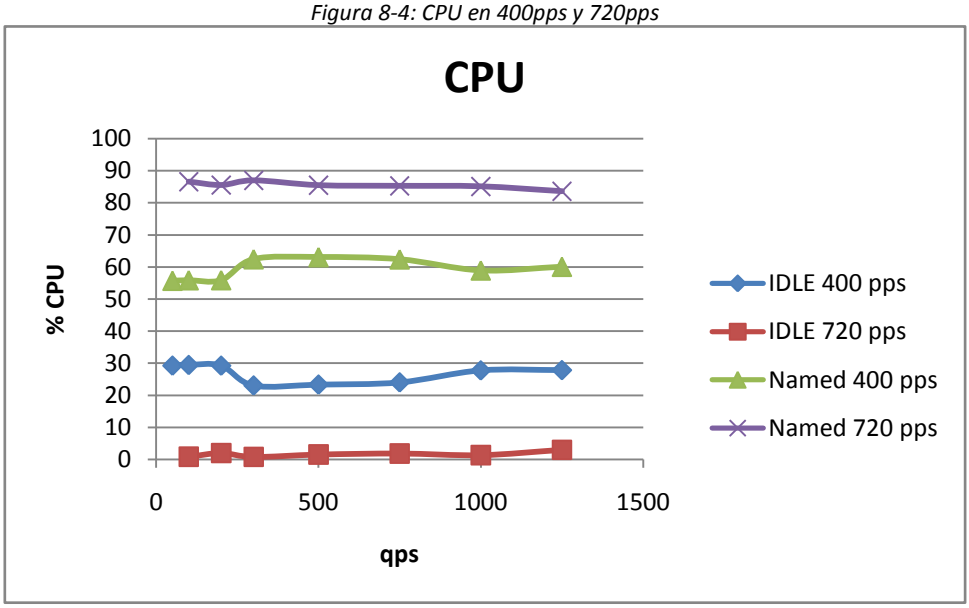
Lo que sí se puede observar de las Gráfica 8-20 que las pérdidas se dividen en forma pareja entre las consultas respondidas desde cache y las que no, ya que el porcentaje de cache siempre se mantiene en el entorno del 70%. Igual se recalca que los picos de pérdidas son mínimos en comparación con otras pruebas que se realizaron.

Por lo que se puede concluir que mientras los slots disponibles para realizar consultas no se llenen, la presencia de delay en la red no presentara un problema mayor para el rendimiento del servidor.

8.4.3.1 - Servidor Recursivo Firmado con Delay

Para esta prueba se siguió la práctica del servidor recursivo firmado con 70% de cache. Las medidas se tomaron en 400 y 720pps siguiendo el razonamiento del servidor sin firmar con delay, donde 400pps es un punto de operación normal, y el 720pps contiene un 5% de perdidas.

Figura 8-4: CPU en 400pps y 720pps



Para los dos puntos de operación se observa que el delay no afecta en grandes rasgos a la performance del CPU, ya que estos varían dentro del rango de CPU del servidor sin delay como se puede observar en Tabla 8-3.

Tabla 8-3: Referencia

	Delay(ms)	Pérdidas %	IDLE	Named
Sin Delay-400pps		0.00	24.50	61.80
Mínimo	100	0.13	29.41	55.87
Máximo	1500	5.40	20.82	64.87
Sin Dely-720pps		3.59	0.22	85.85
Mínimo	100	6.90	0.82	86.56
Máximo	1500	20.24	3.75	81.80

Uno de los problemas de las medidas de CPU, es que estas no son muy exactas, ya que al principio el recursivo solo responde consultas de cache y reenvía consultas a la zona con delay guardando esta en un buffer. Por lo que durante el tiempo de delay el CPU consume menos recursos mientras las respuestas no le lleguen. Cuando comienzan a llegar las respuestas, además de realizar el reenvío de consultas el servidor debe comenzar a autenticar los registros, por lo que el consumo de CPU se incrementará en esta etapa. Debido a esto el promedio brinda un resultado menor de consumo de CPU, siendo más notorio esto a delays superiores a 1000ms.

Figura 8-5: Pérdidas a 400pps

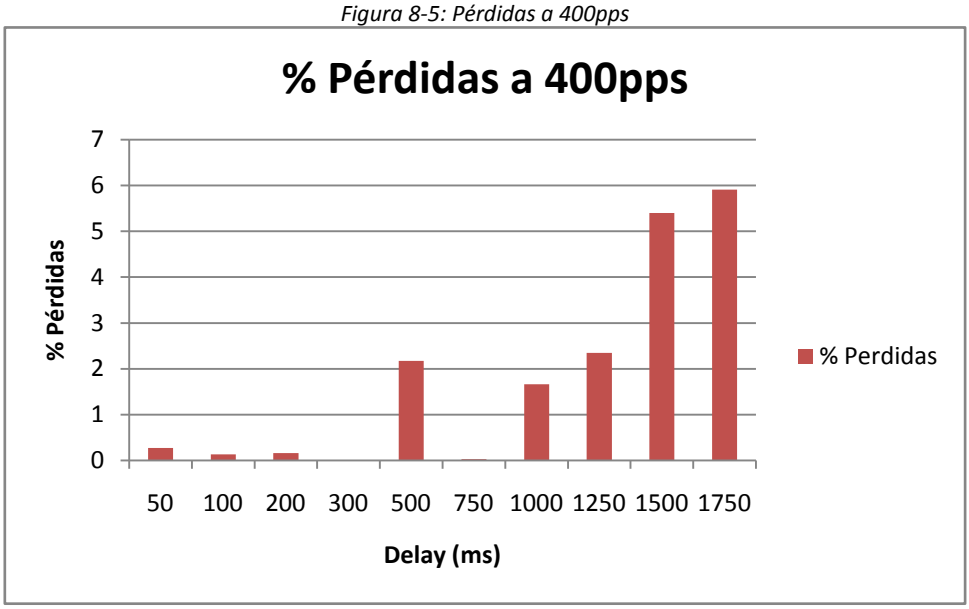
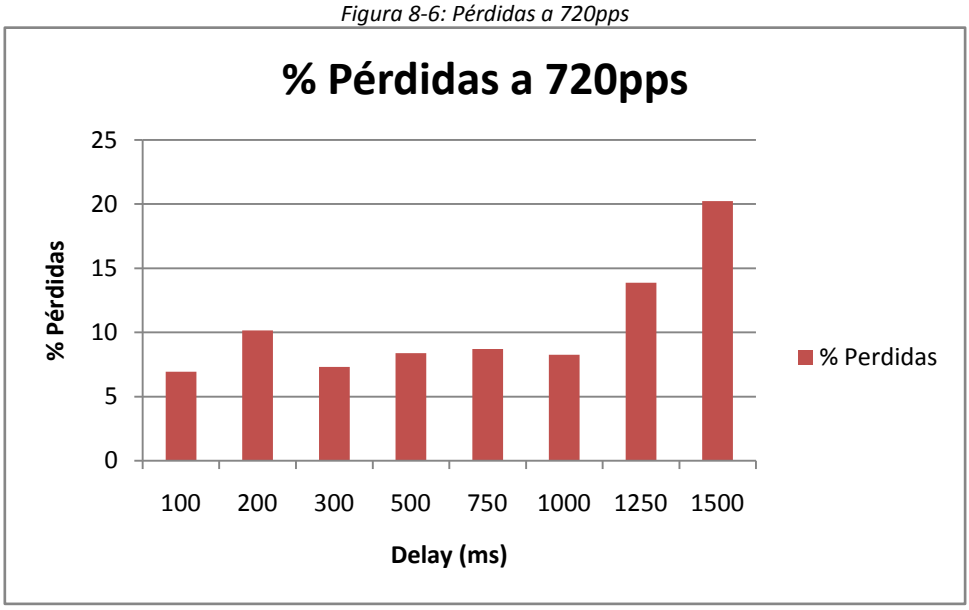


Figura 8-6: Pérdidas a 720pps



Las pérdidas en los dos puntos de operación son más notorias principalmente en tiempos muy grandes, ya que el recursivo debe mantener por 1.5 segundo las consultas sin respuesta en su buffer de salida.

En conclusión, los retados que agregan otros servidores sobre las respuestas, no afectan el consumo de CPU en el servidor firmado y no firmado con respecto a los servidores sin delay. Pero existe un mayor grado de pérdidas en el servidor firmado que en el no firmado, esto se atribuye a que el servidor firmado realiza verificaciones de la cadena confianza consumiendo más CPU. En una red real, es de esperar que en la mayoría de las consultas el retardo de ida y vuelta hacia los servidores no sea mayor de 500 ms, por lo que no esperamos que esto presente algún un problema.

8.5 - NXDOMAIN

Una de las principales decisiones que debe tomar un administrador de zona es si utilizar NSEC o NSEC3 para proveer autenticación de no existencia. Esta decisión es en parte administrativa, respondiendo a la necesidad de evitar la enumeración de zona. Sin embargo, también se debe tener en cuenta el impacto en el consumo de recursos que el uso de NSEC3 genera. Al utilizar NSEC3, tanto el servidor recursivo como el autoritativo deben realizar operaciones con hashes extras −con respecto al uso de NSEC−. Además se requiere del intercambio de más registros NSEC3 que NSEC en la autenticación de no existencias.

8.5.1 - Parámetros NSEC3

Un servidor de nombres autoritativo de una zona firmada debe proveer autenticidad de no existencia cuando éste es consultado por un nombre o registro que no existe. Con NSEC3, el servidor autoritativo debe realizar operaciones de hash para identificar qué registro NSEC3 se debe devolver, mientras que el recursivo debe realizar estas mismas operaciones más las verificaciones de firma. El impacto de estas operaciones en la performance del servidor recursivo estará determinada por la cantidad de consultas NXDOMAIN o NODATA que le llegue y de los parámetros del registro NSEC3 definidos en el RR NSEC3PARAM. Estos parámetros son la cantidad de iteraciones −número de veces que se realiza un hash en el algoritmo− y la salt.

La elección de estos parámetros −en particular del número de iteraciones− depende del tipo de equipo que se tenga disponible para servir la zona. Según pruebas realizadas en el k-root [37], la performance del servidor se degrada rápidamente al incrementar el número de iteraciones. Por lo que para encontrar el valor de este parámetro que mejor ajuste nuestro sistema y nuestras necesidades de seguridad, se debe tener en cuenta la curva qps_máximas vs iteraciones.

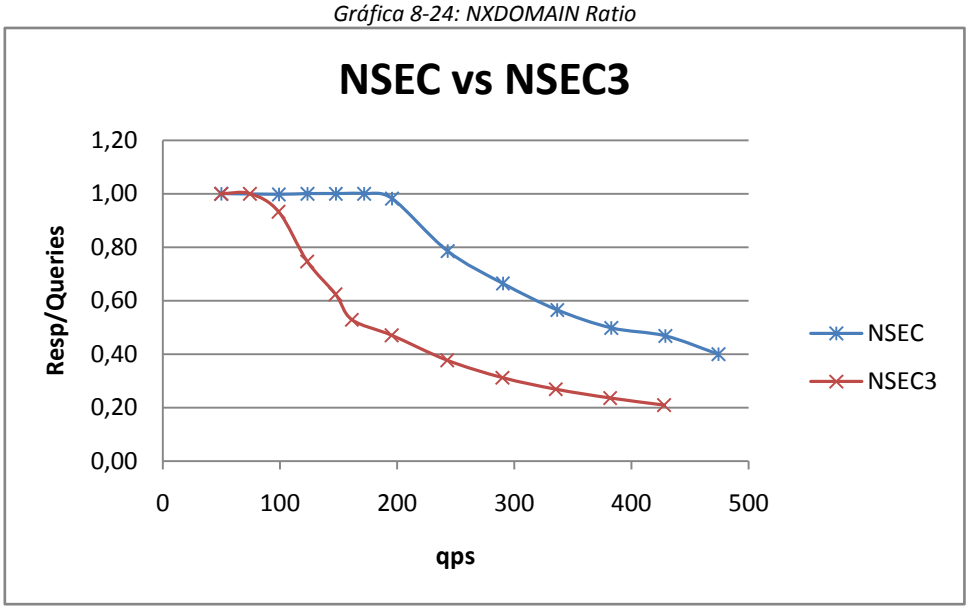
Un estudio interesante encontrado en la web [38], resume los parámetros elegidos por administradores de TLD que optaron por firmar sus zonas con NSEC3. Entre los TLD más importantes tenemos a com, edu y net, de los cuales ninguno utiliza ni iteraciones ni salt para la realización de los RR NSEC3. Por otro lado, dominios como gov, jp −Japón−, y gr −Alemania− utilizan una salt de ente 2 y 6 bytes, y 10 iteraciones, siendo este último el valor por defecto en el BIND. La mayoría de los administradores de zona optaron por un número de iteraciones menor a diez, si bien encontramos algunos operadores que se tomaron muy en serio el problema de enumeración de zona y que utilizan 150 iteraciones.

Para realizar nuestro experimento y determinar la diferencia entre resolver autenticidades de no existencia con NSEC y NSEC3 decidimos dejar fijos los parámetros de NSEC3PARAM en 10 iteraciones y 4 bytes de salt. En particular, variar estos

parámetros requeriría refirmar las zonas cada vez que cambia un valor de estos parámetros.

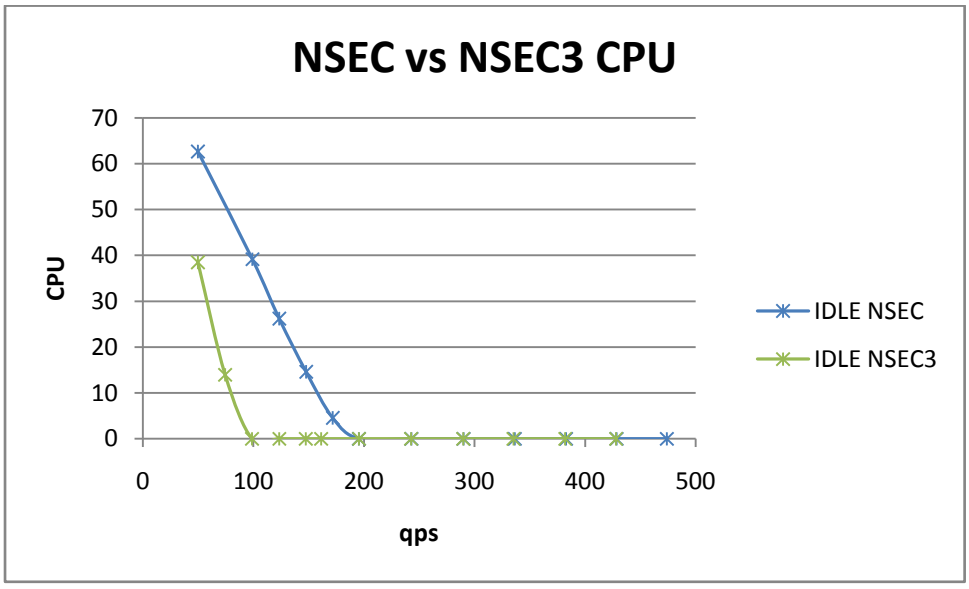
Para generar las consultas utilizamos un programa que genera consultas por registros inexistentes de zonas autoritativas de nuestro servidor. La maqueta utilizada corresponde a la misma utilizada en la prueba de servidor recursivo, Figura 8.3.

Gráfica 8-24: NXDOMAIN Ratio



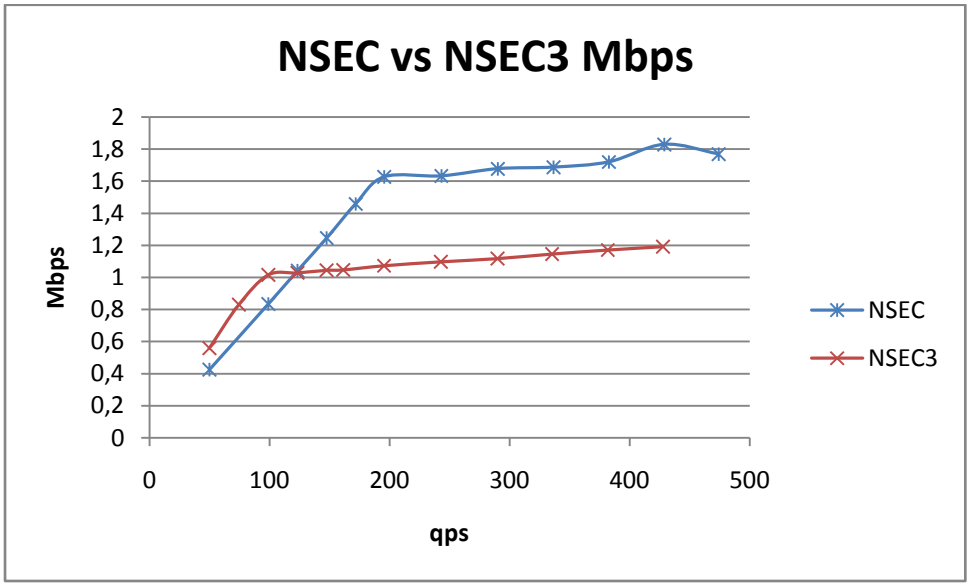
En esta primera gráfica se observa cómo se degrada la performance de nuestro servidor al estar resolviendo una zona firmada con NSEC3. A 200 pps tenemos que NSEC recién empieza a caer, mientras que NSEC3 logra resolver un 100% de las consultas hasta los 75qps.

Gráfica 8-25: NXDOMAIN CPU



Como era de esperar, encontramos que el consumo de CPU es mayor para NSEC3 que para NSEC, esto es debido a la realización de los hashes necesarios para verificar que los registros NSEC3 con correctos, y a la verificación de las firmas digitales de los mismos. Nuevamente encontramos que el CPU es el cuello de botella de nuestro sistema, ya que cuando tenemos 0% de CPU IDLE el ratio comienza a degradarse.

Gráfica 8-26: NXDOMAIN Mbps



Si observamos la Gráfica 8-26 en la zona lineal −antes de que comiencen a caer las curvas−, encontramos que NSEC3 consume un poco más ancho de banda por paquete. Esto se debe a que el servidor autoritativo debe devolver tres RR NSEC3 −y sus firmas− para autenticar la no existencia, mientras que solo se necesitan 2 en NSEC para hacer lo mismo. Cuando ambas curvas saturan −CPU IDLE = 0− tenemos que la relación Mbps/qps es igual −curvas paralelas−. Esto se debe a que tanto NSEC y NSEC3 llegan a un máximo de consultas por segundo que pueden resolver, en el caso de NSEC es de 100, mientras que para NSEC3 es de 200. Esto determina que la curva saturada de NSEC esté por encima. Por otro lado, dado que la cantidad de paquetes que se envían a cada qps es la misma tanto para la prueba de NSEC y NSEC3, resulta que las curvas son paralelas.

9. Recomendaciones

9.1 - Despliegue de DNSSec

El despliegue de DNSSec introducirá nuevos procedimientos y operaciones en la administración de los servidores de nombres autoritativos y recursivos. La administración de claves, que abarca su generación, almacenamiento, uso y renovación, son tareas que deben realizarse periódicamente y bajo responsabilidades bien definidas. El mantenimiento de la cadena de confianza estará a cargo también del administrador de la zona, quien es responsable de la publicación de la clave pública en la zona padre −el RR DS−. Además de todo esto, se recomienda a la organización llevar registro de todos los procedimientos, tareas y de los roles de confianza que se asignan a los individuos, en un documento que especifique todas estas prácticas referentes a la operación de la zona.

Resumiremos aquí los principales delineamientos que se recomienda seguir en el despliegue de DNSSec en una organización, según las indicaciones de instituciones gubernamentales de EEUU y Europa, NIST [39] y ENISA [40] respectivamente, RFCs [23], y operadores de zona y proveedores de software como por ejemplo NLnet Labs [41].

9.2 - Firmar la zona

Lograr tener totalmente operativa una zona firmada conlleva una cantidad considerable de pasos y tiempo que son necesarios para asegurar el correcto funcionamiento y transparencia en la operación de la zona.

9.2.1 - Sistema de firmado

La primera consideración a tomar al diseñar el sistema que firmará las zonas es como éste se integrará a la infraestructura DNS ya existente en la organización. La mejor práctica es hacer uso de un repositorio interno −Configuración *Hidden Master*− que contenga el archivo de zona, y un servidor para la transmisión o envío de estos datos a los demás servidores autoritativos que sirven la zona. Estos servidores deberían utilizar el NOTIFY [42] y AXFR [3] entre alguno de los mecanismos más comunes para las transferencias de zonas.

El sistema responsable del firmado de la zona estará entre el repositorio interno y los mecanismos de entrega de datos al exterior. En la práctica, se debería integrar el sistema de firmado al servidor Hidden Master, de modo que en este servidor se realice la firma de la zona y el mantenimiento de los datos. Las especificaciones de este sistema responsable de las firmas deben ser consideradas tomando en cuenta que éste debe realizar la administración, seguridad de las claves y los datos, y la performance del sistema dado que será sometido a operaciones criptográficas.

El sistema utilizado para firmar, debe ser totalmente configurable, soportando diferentes algoritmos de firmado incluyendo de forma obligatoria RSASHA1 y DSA con tamaños de claves desde 1024 hasta 4096 bits. El período de validez de las firmas y claves debe también poderse configurar. El sistema deberá proveer mecanismos que permitan firmar y cambiar las claves de forma manual y totalmente automática –Anexo F sección 6.1.4 - −, siendo esta última la recomendable para la operación de la zona.

La generación de las claves utilizadas en las firmas digitales y el almacenamiento de las mismas debe estar presente junto con el sistema de firmado. Dependiendo de las necesidades de seguridad de la organización, puede ser necesario el uso de un HSM −Hardware Security Module−. Estos dispositivos deben cumplir con diferentes requerimientos de procedimiento, como la especificación FIPS 140 Nivel 4, y requerimientos funcionales como la generación de números aleatorios −Anexo G sección 7.2 - Estos dispositivos deben ser utilizados en la administración de las claves para generarlas, almacenarlas y hacer uso de ellas. Todas las operaciones criptográficas que requieran el uso de la porción privada de una clave deben realizarse dentro del HSM. La clave no debe salir de este dispositivo a no ser que sea en forma cifrada.

Aunque tengamos operativa nuestra zona firmada con DNSSec, no se pueden descuidar las recomendaciones de seguridad “estándares” de un servidor DNS expuesto en Internet. La seguridad en DNS debe comenzar por asegurar nuestro sistema de firmado, algunas de las medidas recomendadas son las siguientes [39]:

- Correr las últimas versiones del OS y software disponibles, estar informado sobre vulnerabilidades, parches, etc., que estén documentados.
- Limitar la exposición de información acerca del sistema utilizado. No permitiendo que un atacante conozca la versión utilizada del software.

- Restringir el acceso a los servicios. Limitar los clientes que pueden realizar consultas recursivas a un servidor disminuye la probabilidad de un ataque DoS.
- Limitar el daño que un ataque exitoso pueda realizarle al sistema. Permisos de escritura restringidos, división de zonas en diferentes archivos, *views*.

Otro aspecto a tener en cuenta en el diseño del sistema de firmado, es que este debe estar conectado a un servidor UTC. Esto es necesario ya que con DNSSec la nocion de tiempo de un servidor DNS toma una carácter absoluto. Es necesario que todos los servidores DNSSec de la red tengan un mínimo de sincronización temporal, para poder asegurar que las fechas indicadas en los registros RRSIG tengan significado.

9.2.2 - Pruebas del sistema

Siguiendo con las recomendaciones anteriores −para implementar el sistema de firmado de las zonas− es recomendable la realización de pruebas que verifiquen el correcto funcionamiento de todos los equipos en conjunto en un ambiente controlado antes de poner en producción las zonas firmadas. Las pruebas a realizar deben incluir la generación de claves, el firmado, la revocación de claves, el roll-over de claves ZSK y KSK, verificando la disponibilidad de la zona durante estos procedimientos. Todos los procedimientos involucrados deben testearse en un período de prueba, con el propósito de acelerar este período es posible disminuir el ciclo de vida de las claves y firmas sin perder generalidad.

Una vez realizadas todas estas pruebas y adquirida la experiencia y confianza en la operación de DNSSec, el siguiente paso es dejar disponibles las zonas firmadas a los resolvers internos de la organización. Se deberá configurar estos resolvers para que puedan validar datos DNSSec, por lo que estos tendrán configurada un ancla de confianza conteniendo las claves públicas de las zonas a validar. En este paso no es necesaria la publicación del registro DS en la zona padre, ni tampoco recomendada. En esta etapa se debe verificar que el resolver pueda validar correctamente los datos obtenidos del servidor autoritativo así como lograr la actualización automática de la KSK según la RFC 5011 [27]. En esta etapa se conformó lo que se conoce como una Isla de Seguridad, en la que cualquier resolver configurado con un ancla de confianza de las zonas firmadas puede validar los datos. Sin embargo, no es posible la construcción de una cadena de confianza desde zonas superiores.

El segundo paso antes de lograr una zona DNSSec totalmente operativa es verificar que todos los servidores autoritativos externos −generalmente esclavos− soporten DNSSec. Se debe considerar que DNSSec no asegura la transferencia de datos de zona a estos servidores externos, por lo que algún mecanismo complementario debe implementarse para asegurar estas transacciones −si es que ya no lo están−. Por ejemplo TSIG [43] −Transaction Signature− provee los medios para identificar los host que tienen permitido realizar transferencias de zona.

9.3 - Generación de claves

Las claves DNSSec se clasifican en dos categorías: las ZSK −Zone Signing Keys− utilizadas para firmar los datos de la zona y las KSK −Key Signing Keys− utilizadas para firmar los registros de las claves −DNSSKEY−. Estas últimas son las utilizadas para formar la cadena de confianza y tendrán una vida útil más larga que las ZSK. Las claves KSK también se caracterizan por ser más largas, aunque esto no afecte la performance del sistema en el firmado o validación de firmas, ya que estas solo se utilizan para firman los registros de las claves ZSK y KSK.

El largo de las claves depende del algoritmo seleccionado y de la vida útil de la clave. Las recomendaciones actuales [23], indican el uso de RSASHA1 con tamaños de claves mayores o iguales a 2048 bit de longitud y 2 años de vida útil para la KSK y 1024 bit de longitud y tres meses para la ZSK. También se recomienda considerar el uso de RSASHA256 dadas las vulnerabilidades encontradas en SHA1 −colisiones de hash−.

Dada la importancia de las claves −en especial las KSK utilizadas para construir la cadena de confianza−, la generación de las mismas estará a cargo de personas que cumplan un rol de confianza. La generación de claves para las zonas de alto nivel como las TLD debe tomar lugar durante una ceremonia en la que participen personas ajenas a la organización y que garanticen la transparencia de las operaciones, documentando todos los procedimientos de forma de avalar la confianza depositada en dichas claves.

El almacenamiento de las porciones privadas de las claves, tanto de la KSK como la de la ZSK, dependerá si se usa DNS dinámico o no. En caso negativo se debe separar la unidad de almacenamiento de dichas claves del servidor autoritativo maestro. Si de lo contrario se hace uso de DNS dinámico, las claves privadas correspondientes a la ZSK deben estar disponibles en todo momento en dicho servidor, pero se recomienda que estén protegidas en un HSM.

La decisión sobre si utilizar NSEC o NSEC3 para proveer autenticidad de no existencia debe estar acompañada por estudios de impacto en la performance de los servidores. Dado que NSEC3 requiere mayor ancho de banda en comparación con NSEC y debe realizar operaciones criptográficas en tiempo de operación, se debe dimensionar el sistema en función del tráfico que recibe y el porcentaje de respuestas NXDOMAIN que envía. Zonas conteniendo pocos nombres de dominio, que además sean predecibles − como www, ns1 y ftp− y cuyos dueños no presenten reparos frente a la enumeración de zona, deben utilizar NSEC. De lo contrario, administradores que deseen brindar un plus de seguridad y utilicen NSEC3 deben elegir los parámetros de estos registros −iteraciones y salt−. La performance del sistema que firma y sirve estas zonas estará relacionada con la elección de estos parámetros, se recomienda [44] el uso de una sola iteración y 8 bytes de salt, la cual se debe modificar junto con los roll-over de las firmas.

Un operador que administre varias zonas puede optar por utilizar la misma KSK y ZSK para firmar todas las zonas bajo su administración. Hacer esto no disminuye la seguridad de ninguna zona ni debilita la resistencia de las claves frente a ataques. Sin

embargo, usar las mismas claves para diferentes zonas incrementa el impacto de un ataque ya que varias zonas son comprometidas en caso de que el ataque resulte exitoso.

Las claves KSK son las que tienen un mayor valor, tanto para el dueño como para un posible atacante. Por lo que estas deben ser generadas, almacenadas y accedidas desde un HSM que proteja la clave frente a accesos no autorizados, entre otras cosas.

9.3.1 - Roll-over de Claves

Los mecanismos para los roll-over de las KSK y ZSK deben estar diferenciados, al igual que sus calendarios. Si bien DNSSec no hace esta distinción, en la práctica es necesaria debido a que el roll-over de la KSK, a diferencia del de la ZSK, precisa de la interacción con la zona padre. El administrador de zona debe conformar un calendario conteniendo la planificación de estos roll-over diferenciados, los cuales se recomienda sean automatizados. También se debe incluir la posibilidad de un roll-over de emergencia −de forma manual− de cualquier clave en cualquier momento del calendario. Los roll-over de emergencia deben realizarse cuando se sospecha de la integridad de una clave.

Es recomendable para el roll-over de la ZSK utilizar el método de pre publicación, con un tiempo de introducción aproximado de 4 días. El tiempo de introducción es el lapso de tiempo desde que se publica la nueva clave en la zona hasta que se puede utilizar activamente. Este tiempo depende de la propagación de los RR y está determinado por los TTL de los registros y los parámetros del SOA. Un roll-over planificado de estas claves es sencillo y totalmente automatizable −ver Anexo F−, permitiendo que la zona permanezca operativa durante este procedimiento. Un roll-over no planificado −de emergencia− se debe realizar si se sospecha de la integridad de la clave, y debe realizarse inmediatamente de modo de minimizar la probabilidad de un ataque. En este caso, si existe una clave en espera −y cumplido el tiempo de introducción− lista para ser activada, este roll-over puede realizarse sin impacto alguno en las operaciones del servidor.

El roll-over de la KSK se diferencia del de la ZSK por precisar interactuar con la zona padre. Estos roll-over pueden afectar la cadena de confianza ya construida, por lo que la ejecución de este proceso debe llevarse con mayor atención. Se recomienda el uso de la doble firma para la realización del roll-over. La zona hija, quien desencadena este roll-over deberá enviar de forma segura la nueva clave pública KSK a la zona padre, implementando algún método de seguridad en el canal o que permita validar autenticidad de la misma. Si se trata de un roll-over planificado se puede utilizar la cadena de confianza ya establecida −por la KSK que se quiere cambiar− para validar la nueva clave KSK. Si de lo contrario estamos tratando con un roll-over de emergencia, debido a la pérdida o ataque a la KSK en uso, no es posible hacer uso de la cadena de confianza ya establecida para autenticar el roll-over. En este caso es necesario hacer uso de métodos fuera del protocolo DNSSec para autenticar la nueva clave.

El roll-over de una KSK que es utilizada como ancla de confianza debe tener un periodo de introducción mínimo de 30 días [27], tiempo durante el cual debe ser siempre validada por una clave de confianza ya establecida. Transcurrido ese tiempo la nueva clave de confianza ya puede utilizarse para crear una cadena de confianza y validar datos.

9.4 - Operación de servidor de nombre recursivo

Se debe tomar en cuenta al dimensionar un servidor recursivo DNS los picos de consumo de CPU del servidor en los primeros momentos de operación, en los que se asemejará más a un servidor sin cache. Debe tenerse en cuenta que en este tiempo es donde el consumo de CPU y ancho de banda es máximo y el servidor es más vulnerable frente a un ataque. Un ataque DoS puede aprovecharse si un servidor no restringe los clientes a los que les ofrece recursividad. Por lo que es una buena práctica restringir las redes −direcciones IP− que tienen permitido enviar consultas recursivas a los servidores.

10. Conclusiones

Conclusiones Generales

El presente trabajo tuvo como primera contribución el estudio en profundidad de DNSSec, incorporando una síntesis de este nuevo mecanismo de seguridad del DNS. Se realizó una recopilación de material bibliográfico que incluye las RFC relevantes a DNSSec, documentación y papers conteniendo no solo información sobre este tema, sino también resultados prácticos utilizados como guía para nuestras propias pruebas.

De las pruebas realizadas en este proyecto y el análisis realizado durante el estudio de estas extensiones de seguridad, se extraen recomendaciones y delineamientos a seguir en el despliegue de DNSSec en una organización. Se explican también cuales son los compromisos existentes en el diseño de un sistema que sirve DNSSec y que un administrador debe tomar en cuenta para operar una zona firmada.

Conclusiones de la maqueta

Las pruebas realizadas durante el proyecto se enmarcaron bajo la consigna de verificar lo estudiado sobre DNSSec. Se encontraron resultados concluyentes respecto a la performance de los servidores DNSSec, verificándose los resultados esperados y documentados en la bibliografía.

La utilización de las trazas reales de SeCIU nos permitió concluir que los resultados obtenidos con nuestros equipos pueden ser aplicados a otros sistemas y organizaciones que manejan niveles de tráfico DNS semejantes al de SeCIU. Esto es importante como referencia para dichas organizaciones que deseen desplegar DNSSec y necesiten redimensionar sus servidores.

Dentro de toda la infraestructura del DNS, encontramos que los servidores de nombre autoritativos son los que menos se ven impactados por el despliegue de DNSSec. Si en el servidor se operan zonas firmadas utilizando NSEC como mecanismo de autenticación de no existencia, este servidor solo tendrá que devolver una mayor cantidad de recursos que antes −en particular los RRSIG−. Si de lo contrario se utiliza

NSEC3 como alternativa en la autenticación de no existencia, la performance de los servidores autoritativos estará limitada por la cantidad de respuestas del tipo NXDOMAIN y NODATA que debe devolver. Siendo estas las responsables de las operaciones criptográficas necesarias en las respuestas NSEC3, que degradan seriamente la performance del servidor.

Las pruebas realizadas sobre un servidor de nombre recursivo mostraron una degradación mucho más importante que la encontrada en un servidor autoritativo. Además de las resoluciones de consultas recursivas que estos servidores ejecutan, también deben realizar las operaciones criptográficas necesarias para validar no solo los registros sino también las cadenas de confianza. En caso de que las consultas devuelvan registros NSEC3, también se deben agregar las operaciones necesarias para validar la construcción de dichos registros.

Consideramos que estos resultados pueden ser valiosos tanto para comparar los tipos de operaciones de los servidores como para los operadores de servidores recursivos que deben enfrentar el despliegue de DNSSec. Estos servidores cumplen un papel muy importante, ya que es en estos servidores donde se realizará la mayor cantidad de validaciones DNSSec por parte de sus clientes.

En conclusión, la implementación de DNSSec presenta un reto importante para las empresas y organizaciones interesadas en mejorar la seguridad de sus servicios en Internet. El aumento de los recursos necesarios para servir y resolver datos de zonas firmadas puede llevar a dichas organizaciones a re-evaluar su infraestructura. Como se vio en las pruebas el impacto en la performance de los servidores puede ser considerable, sin embargo, se estudió el peor caso −o el mejor, en el sentido de un despliegue exitoso de DNSSec− en el que todas las consultas están firmadas.

Trabajo futuro

- Estudio del comportamiento de servidores implementando nuevas herramientas, que nos permitan la realización de medidas más exactas.
- Se plantea la realización de una maqueta distinta donde el cuello de botella sea la red o el buffer en caso de servidores recursivos.
- Realizar un seguimiento de lo realizado por el SeCIU, cuando este implemente DNSSec.
- Realizar medidas de performance en servidores en producción, teniendo en cuenta, los roll-overs y actualizaciones dinámicas de zonas −DDNS−.
- Analizar el impacto de DNSSec desde el punto de vista del cliente. Teniendo en cuenta, en particular, el delay que se adicionará a una respuesta DNS debido a las operaciones critográficas involucradas en la validación de los RR.

Siglómetro

AD	Authenticated Data
CD	Checking Disabled
BIND	Benkeley Internet Name Domain
CA	Certificate Authority
CCTLD	Country Cod Top Levels Domians
DDNS	Dynamic DNS
DLV	DNSSec Look aside Validation
DNS	Domain Name System
DNSSec	Domain Name System Security Extensions
DO	DNSSec OK
DoS	Denial of Service
EDNS	Extension mechanisms for DNS
FQRN	Fully Qualified Domain
gLTD	Generic Top Level Domain
HSM	Hardware Security Module
IANA	Internet Assigned Numbers Authority
ICANN	Internet for Assigned Numbers and Named
IDEL	Tiempo ocioso
KSK	Key Signing Key
NIC	Network Information Center
NIST	National Institute of Standards and Technology
NITA	National Telecommunications and Information Administration
PID	Process ID
PKI	Public Key Infrastructure
PNE	Prueba de No Existencia
PPS	Packet Per second
RR	Resource Record
RR DNSKEY	Resource Record DNS Public Key
RR DS	Resource Record Delegation Signing
RR NSEC	Resource Record Next Secure
RR RRSIG	Resource Record Signature
RAU	Red Académica Uruguaya
RRset	Resource Record Set
RSA	Rivest, Shamir y Adleman
SeCIU	Servicio Central de Informática de la Universidad de la Republica
SEP	Secure Entry Point
SHA	Secure Hash Algorithm
SLD	Second Level Domain
SSL	Secure Socket Layer
TLD	Top Level Domain
TSIG	Transaction SIGnature
ZSK	Zone Signing Key

1. Anexo A

1.1 - Formato del Archivo de Zona

Si bien el contenido de un archivo de zona está estandarizado en la RFC1035 [4], su formato puede variar según su implementación. Aquí describiremos el formato utilizado por BIND, que fue el programa utilizado durante el proyecto.

Los archivos de zona son archivos de texto que se pueden leer o editar con cualquier editor estándar y puede contener tres tipos de entradas:

- ***Comentarios:*** Todos los comentarios comienzan con un punto y coma, continuando hasta el final de la línea. Los comentarios se pueden añadir a cualquier tipo de registro.
- ***Directivas:*** todas las directivas comienzan con un signo de pesos −$− y se utilizan para controlar el procesamiento de los archivos de zona.
- ***Registros de recursos:*** Los registros de recursos −RR− se utilizan para definir las características, propiedades o entidades que figuran dentro del dominio. Los RR están contenidos en una sola línea con la excepción de que las entradas entre paréntesis se pueden propagar a través de varias líneas.

Los separadores de campo de un RR pueden ser espacios o tabulaciones. En los archivos de zona, los tabuladores se utilizan tradicionalmente para hacer un diseño más atractivo e indicar claramente los campos faltantes.

Ejemplo 1-1: Fragmento de un Archivo de Zona

```
; Esta es una línea completa de comentarios
$TTL 12h ; Directiva - Comentario termina la línea
$ORIGIN ejemplo.com.
; Inicio de Autoridad (SOA) que define la zona(o dominio)
; ilustra un registro RR distribuido en más de una línea
; utilizando los paréntesis
@ IN SOA ns1.ejemplo.com. hostmaster.ejemplo.com. (
                                          2003080800 ; se  = serial number
                                          3h         ; ref = refresh
```

```
                                        15m         ; ret = update retry
                                        3w          ; ex  = expiry
                                        2h20m       ; min = minimum
                                        )

; Única línea de RR
   IN NS ns1.ejemplo.com.
...
El RR Inicio de Autoridad (SOA) anterior pudo ser escrito en una sola línea de
este    ; modo
 @ IN SOA ns1.ejemplo.com. hostmaster.ejemplo.com. 2003080800 3h 15m 3w 3h
```

1.1.1 - Contenido del Archivo de Zona

Un archivo de zona contiene los siguientes RR y directivas, aquí daremos una rápida descripción sobre sus usos y funciones, más información se puede encontrar en [45]:

- *La directiva $TTL*: Define el valor Time to Live de la zona o de dominio, es el tiempo que un RR puede ser almacenado en cache −o salvado− por otro servidor DNS. Esta directiva es obligatoria en caso de que existan RR que no posean este campo.
- *La directiva $ORIGIN*: Es el nombre de dominio de la zona que se está definiendo. Esta directiva es opcional.
- *Inicio de Autoridad −RR SOA−:* Éste debe aparecer como el primer RR en un archivo de zona, describiendo las características globales de la zona o dominio. Sólo puede haber un RR SOA en un archivo de zona, este RR es obligatorio.
- *Servidor de nombres −RR NS−*: Define los servidores de nombres que tienen autoridad para la zona o dominio. Se recomienda el uso de dos o más servidores de nombre, cada uno descrito en su propio RR NS.
- *Intercambiador de Correo −RR MX−*: Define los servidores de correo para la zona. Puede haber cero o más MX en un archivo de zona. Si un dominio no ofrece servicios de correo electrónico, no hay necesidad de RR MX. Al igual que el RR NS, el MX puede hacer referencia a un servidor correo en este dominio o a uno externo. Este RR es opcional.
- *Dirección −RR A−*: Se utiliza para definir la dirección IPv4 de los host −o servicios− que existen en una zona y están obligados a ser públicamente visibles. Las entradas de IPv6 se definen mediante el RR AAAA −llamado Quad A−. Puede que cero o más registros de A o AAAA estén dentro de un archivo de zona. Este RR es opcional.
- *Nombre Canónico −RR CNAME−*: Define un alias para un registro de recurso, que permite a un host −o servicio− ser definido con el alias de otro host. Puede haber cero o más registros de recursos CNAME en un archivo de zona. Este RR es opcional.
- Existen también otros registros de recursos los cuales se encuentran detallados en el Anexo B.

Los RR anteriores y directivas permiten la definición de un archivo de zona en pleno funcionamiento.

1.1.1.1 - Directivas $TTL

Cada RR puede tomar un valor Time to Live opcional especificado en segundos. Ésta define el valor por defecto del TTL que es aplicado a cualquier RR que no posea un TTL definido. TTL en el contexto del DNS define el tiempo en segundos que un registro puede ser almacenado en cache por otro servidor de nombres o resolver. El TTL es asignado por el administrador de la zona donde se origina el dato. Mientras que los TTL cortos se utilizan para minimizar el cache y cero para prohibir el cacheo, la búsqueda de eficiencia sugiere que esos tiempos sean de días para host típicos de Internet. Si se debe realizar algún cambio, es una buena práctica reducir el TTL para minimizar inconsistencias durante el cambio, y después del mismo se lo incrementa nuevamente a su valor original.

Ejemplo 1-2: TTL

```
$TTL time-in-seconds

$TTL 2d.
```

La directiva anterior $TTL, especifica un valor de TTL de dos días para todos los RR que no tengan definido explícitamente el valor de este campo. El programa BIND permite hacer uso de letras, distinguiendo entre mayúsculas y minúsculas, para simplificar la lectura de estos tiempos. Los caracteres en minúscula permitidos son m=minutos, h = horas, d = día, w = semana. El tiempo en segundos puede tomar el valor 0, el cual significa que nunca se cachea el registro, hasta un máximo de 2147483647 que es más de 68 años. La actual recomendación práctica [46] propone un valor de más de un día, y en los RR que rara vez cambian debe considerarse valores de varias semanas.

TTL determina dos características operacionales de DNS:

1. *Carga de acceso −Access load−*: Cuanto menor es el TTL más rápido se quita del cache, obligando a que se produzcan consultas DNS con más frecuencia y por lo tanto eleva la carga operativa en el servidor de nombres de la zona.
2. *Cambio de propagación −Change propagation−:* El valor TTL representa el tiempo máximo que le toma propagar cualquier cambio desde el servidor de nombre de la zona a todos los usuarios.

El uso de la directiva $TTL puede simplificar la modificación de los TTL del archivo de zona. Cambiando este valor, el administrador puede modificar el valor TTL de todos los RR que estén utilizando la directiva −los que no tienen definido explícitamente el TTL en su RR−.

1.1.1.2 - Directiva $ORIGIN

La directiva $ORIGIN fue estandarizada en la RFC 1035 [4], describe el nombre de dominio que será añadido a un nombre incompleto −a veces llamado nombre no calificado− definido en un RR.

Regla $ORIGIN de Sustitución: Si un nombre aparece en un RR y no termina con un punto, entonces el valor de la última o única directiva $ORIGIN se añadirá al nombre. Si el nombre termina con un punto, entonces es un nombre de dominio completo −FQDN− y no se añadirá nada. El punto terminal en un FQDN se interpreta como la raíz del árbol de dominio o jerarquía. Las directivas $ORIGIN pueden aparecer en cualquier parte de un archivo de zona. No son obligatorias.

Ejemplo 1-3: Directiva Origin

```
$ORIGIN domain-name
```

2. Anexo B

2.1 - Registro de Recursos del DNS

El comienzo de la línea es el propietario del RR. Si la línea comienza con un espacio en blanco se asume que el propietario es el mismo que el del RR anterior. Las líneas en blanco a menudo se incluyen para una mejor visualización.

Después del propietario va el TTL, el tipo y la clase del RR. La clase y el tipo utilizan la mnemotécnica de antes. Los datos de recursos de la sección RDATA de un RR dado utilizan la representación típica de estos datos.

Ejemplo 2-1: Representación de los RR de un mensaje

```
example.org.         MX      1 mail.example.org.
                     MX      2 mail2.example.org.
mail.example.org.    A       128.9.8.155
                     A       10.1.8.33
mail2.example.org.   A       10.2.8.34
                     A       128.9.8.126
```

El ejemplo anterior muestra seis RRs, con dos RRs para cada uno de los tres nombres de dominio.

Ejemplo 2-2: RR de diferentes clases

```
subdominio.example.com.    IN      A       10.0.0.154
                           CH      A       example.com. 2420
```

El Ejemplo 2-2 muestra dos direcciones para example.com, cada una con dos clases distintas.

2.1.1 - Registro de Recursos SOA

El registro de recursos SOA define las características claves y los atributos para la zona o dominio, y está normalizado en la RFC1035 [4]. En el Ejemplo 2-3 se muestra

la expresión de este RR, y en la Tabla 2-1 explicamos la función de cada parámetro de este registro.

Ejemplo 2-3: Registro SOA

```
name  ttl  class  rr  name-server  e-mail  sn  refresh  retry  expiry  min
```

Tabla 2-1: RR SOA

Sintaxis	Descripción
Name	Nombre de la zona. Generalmente se utiliza @ para hacer uso de la directiva $ORIGIN.
TTL	Si no hay ningún valor TTL definido para el RR, se utilizará el valor definido directiva $TTL, que deberá estar presente en dicho caso.
Class	Define la clase IN –Internet– valor por defecto si se omite. Existen otros valores, pero son poco utilizados.
Name Server	Define el nombre del servidor maestro principal para la zona y tiene un significado especial solo cuando es utilizado con configuraciones dinámicas de DNS. El servidor de nombres definido aquí hace necesaria la definición de este mismo en un RR NS. En las especificaciones de DNS a esto se llama campo MNAME.
e-mail	Define una dirección administrativa de correo para la zona. En las especificaciones de DNS esto se conoce como el campo RNAME.
S/N	Define el número de serie que actualmente está asociado con la zona. El número de serie debe ser actualizado cada vez que se realiza cualquier cambio en el dominio. El valor S/N puede tomar cualquier número en el rango de 0 a 4294967295. Por convención, se utiliza un formato de fecha yyyymmddss, dónde yyyy representa el año en cuatro dígitos, mm es el mes, dd es el día, y ss es el número de secuencia en caso de que archivo de zona se actualiza más de una vez por día. Este valor es utilizado durante las operaciones de trasferencia de zona para determinar si el archivo de la zona ha cambiado. El empleo de la convención de fecha se ha diseñado para minimizar los errores, así como para proporcionar una forma sencilla de seguimiento de la última modificación de la zona, pero no es aplicado universalmente.
Refresh	Es valor refesh indica cada cuanto tiempo en segundos debe un servidor esclavo verificar si hubo o no cambios en el servidor maestro. Cumplido ese tiempo, el esclavo lee el valor de S/N en el RR SOA del servidor maestro y lo compara con el que tiene almacenado. Si el primero es mayor, una operación de transferencia de zona se iniciará para actualizar o volver a cargar la copia de los registros de la zona al esclavo. Dependiendo de cómo se implementan las transferencias de zona, el valor de este parámetro puede determinar con qué rapidez se propagan los cambios desde el maestro al

	esclavo. Los valores típicos de transferencia de zona son de 3 a 24 horas.
Retry	Define el tiempo en segundos entre reintentos. Si el esclavo no puede hacer contacto con el maestro de la zona en un ciclo de refesh, esperará retry segundos antes de intentar devuelta. Los valores típicos son de 10 a 60 minutos.
Expiry	Define el tiempo en segundos después del cual, los registros de zona se asumen de no autoridad. Esto significa que los registros no pueden ser considerados válidos y en consecuencia un servidor deja de responder a las preguntas de la zona. De modo que, cuando el tiempo límite de refresh es alcanzado el esclavo trata de ponerse en contacto con el maestro de la zona, en el caso de una falla, reiterará la reconexión cada retry tiempo. Si el contacto es logrado, tanto el contador refresh como el expiry son reiniciados. Si el esclavo no ha logrado hacer contacto cuando el tiempo expiry es alcanzado, el esclavo dejará de responder a cualquier pregunta. La zona esta esencialmente muerta en este punto. Para permitir apagones importantes expiry se establece normalmente en un valor muy alto, como una o tres semanas.
Nx	Nx fue redefinido en la RFC 2308 [19] como el período de tiempo en que las respuestas negativas pueden ser cacheadas por un servidor. Por lo tanto, si se realiza una solicitud para subdominio.ejemple.com y este no se puede resolver −porque no existe−, entonces el servidor devolverá Name Error −NXDOMAIN−. EL servidor recursivo continuará devolviendo este valor hasta que expire el Nx, momento en el que volverá a intentar la operación fallida.

2.1.2 - Registro de Recursos NS

El registro de recursos NS está estandarizado en la RFC 1035 [4] y define los servidores de nombres autorizados −de los cuales es recomendable que haya al menos dos− para cada zona.

Ejemplo 2-4: RR NS

```
;name   ttl  class   rr  name

; Registro de recurso NS para el dominio
  example.com          IN NS ns1.example.com.
; El Segundo servidor de nombres es
; externo a la zona (dominio).
  example.com          IN NS ns2.example.net.
```

La Tabla 2-2 describe la sintaxis formal para el primer registro NS, descripto en el Ejemplo 2-4, que es interno a la zona.

Tabla 2-2: NS RR Sintaxis

Sintaxis	Descripción
Name	Este campo puede ser un espacio o un carácter de tabulación que sustituye el valor actual del campo de nombre.
TTL	Si no hay ningún valor TTL definido para el RR, será utilizado el valor de la directiva $TTL.
Class	Define la clase a la que pertenece el RR, generalmente IN –Internet–.
Name	Define un servidor de nombres de autoridad para el dominio. Este RR NS apunta al nombre de dicho servidor y por lo tanto debe tener el correspondiente RR A –o RR AAAA si es IPv6– definido.

Se recomienda definir un segundo servidor de nombre para el dominio, para el caso de que un servidor de nombres no esté disponible –por apagones u otro tipo de cortes en el sistema–. El segundo RR NS que es definido en el ejemplo, está ubicado en una zona externa y por lo tanto no precisa tener asociado un registro glue –RR A– ya que el nombre de domino no pertenecerá a la zona en cuestión y puede resolverse buscando otra parte del árbol de nombres.

2.1.3 - Registro de Recursos MX

El RR MX está estandarizado en el RFC 1035 [4] y define los servidores de correo –o intercambiadores de correo electrónico– para el dominio o zona.

Ejemplo 2-5: RR MX

```
;name   ttl   class   rr   preference   name

; Registro de Recurso MX para la zona
        3w    IN      MX    10 mail.ejemplo.com.
; Segundo servidor de mail es externo a la zona
              IN      MX    20 mail.ejemplo.net.
```

La Tabla 2-3 especifica la sintaxis formal para el primer registro MX utilizado en el archivo de ejemplo, que es interno en el dominio.

Tabla 2-3: RR MX

Sintaxis	Descripción
Name	Nombre del dominio al cual está asociado. Si se deja en blanco, se utiliza implícitamente el nombre del RR anterior.
TTL	El uso explicito del TTL en un RR invalida el valor por defecto –definido en la directiva $TTL–. Es poco probable que el dominio RR MX cambie.
Class	Clase a la que pertenece este RR. Generalmente IN.
Preference	Indica la preferencia relativa o prioridad del servidor de correo que se define y puede tomar cualquier valor entre 0 y 65535. Cuanto menor sea el número mayor la preferencia del servidor.
Name	Define un servidor de correo con el valor de preference definido para el dominio.

Clásicamente se define un servidor de correo de respaldo, que tiene un valor de preferencia más bajo –un número más alto–. En el caso de que el primer servidor de correo no está disponible, el servidor de correo de respaldo será utilizado –nuevamente es ideal que este esté separado geográficamente del primero–.

El segundo RR MX se define para estar en un dominio extranjero o externo y por lo tanto no requiere de un RR A si es IPv4 oIPv6 si es RR AAAA.

2.1.4 - Registro de Recurso A

El RR A está estandarizado en la RFC 1035 [4] y define la dirección IPv4 de un determinado host en un dominio o zona. El equivalente para IPv6 es el RR AAAA y será descrito más adelante.

Ejemplo 2-6: RR A

```
;name   ttl   class   rr   ipv4

ns1     IN    A     192.168.254.2
Mail    IN    A     192.168.254.4
www     IN    A     192.168.254.7
```

La Tabla 2-4 muestra la sintaxis formal del primer RR A utiliza ejemplo.

Tabla 2-4: RR A

Sintaxis	Descripción
Name	Nombre del host o servidor. Se puede utilizar el formato FQDN o ser un nombre no calificado, provocando la sustitución de $ORIGIN.
TTL	Si no hay ningún valor TTL definido para el RR, será utilizado para la zona el valor por defecto 2d en la directiva $TTL.
Class	Clase a la que pertenece el RR.
Ipv4	Define la dirección de red IPv4 del host.

2.1.5 - Registro de Recurso AAAA

El RR AAAA es la recomendación actual IETF para la definición de forward mapping de direcciones IPv6 y se define en el RFC 3596 [47]. Es equivalente al RR A utilizado para el forward mapping IPv4:

Ejemplo 2-7: RR AAAA

```
name ttl class rr ipv6
ns1 IN AAAA 2001:db8:0:1::1
```

Recordemos que IPv6 proporciona direcciones según el alcance –local, global–. Los host tendrán direcciones Link Local IPv6, así como direcciones Global Unicast. Cuando el software en un host quiere acceder a un host local, no utiliza un servidor de nombres para buscar la dirección; en cambio utiliza un grupo local Mulicast para encontrar todos esos hosts locales. Solamente direcciones Global Unicast deben aparecer en el archivo de zona.

Tabla 2-5 Sintaxis del RR AAAA

Sintaxis	Ejemplo	Descripción
name	ns1	El nombre no está calificado, haciendo que el valor de la Directiva $ORIGIN sea sustituido. Esto puede ser escrito como ns1.example.com. Utilizando el formato FQDN que puede ser más comprensible.
ttl		No existe un valor TTL definido para el RR, por lo que por defecto en la zona se utilizará 2d en de la directiva $TTL.
class	IN	Define la clase a ser Internet
IPv6	2001:db8:0:1::1	Esta es una dirección Global Unicast. La dirección mostrada utiliza la función de eliminación de cero de IPv6, y podría haber sido escrita como 2001: db8: 0:1:0:0:0:1. El valor de 2001:db8:0 es el global routing prefix asignado por la IANA y los agregadores –los registros de Internet–. El primer 1 es la subred y el valor ::1 es el asignado localmente –usuario final–.

2.1.6 - Registro de Recurso PTR

El RR Puntero −RR PTR− se utiliza para mapear nombres de dominio a diferentes partes del espacio de nombres. En la práctica, se utiliza para la resolución inversa de direcciones IPv4 e IPv6, en comparación con los registros A y AAAA que asignan a un nombre a una dirección IPv4 e IPv6 respectivamente. Las direcciones IPv4 se mapean bajo la zona in-addr.arpa, mientras que las IPv6 lo hace bajo la zona ip6.arpa. La sintaxis formal es la siguiente:

Ejemplo 2-8: RR PTR IPv6 e IPv4

```
Name    ttl    class    rr    name

0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.ip6.arpa    IN   PTR   ns1.example.com.
1.0.168.192.in-addr.arpa.          IN   PTR ns2.exampe.com.
```

La siguiente tabla se mapea la sintaxis formal de un RR PTR utilizado como ejemplo:

Tabla 2-6 Sintaxis RR PTR

Sintaxis	Ejemplo usado	Descripción
Name IPv6	1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0	Esta es la Id de la subred y la ID de interfaz parte de la dirección IPv6 escrita en formato de nibble inversa. Aunque esto parezca un número, es tratado como un nombre. El nombre no está calificado, haciendo que el valor de la directiva $ORIGIN sea sustituido. Esto puede ser escrito como 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6. ARPA. −Utilizando el formato FQDN punto de terminación−, sí este es más comprensible.
Name IPv4		En caso de estar utilizando IPv4, el nombre de domino del RR PTR en la zona in-addr.arpa correspondiente a la dirección 192.168.0.1. será 1.0.168.192.in-addr.arpa.
Ttl		No existe un valor TTL definido para el RR, por lo que por defecto en la zona se utilizará 2d en de la directiva $TTL.
class IN		IN define la clase a ser Internet
Name	ns1.example.com.	Define que una consulta de 2001:db8::1 retornara ns1.example.com.

2.1.7 - Registro de Recursos CNAME

El RR CNAME está estandarizado en la RFC 1035 [4] y define un alias para un host ya existente definido por un RR A.

Ejemplo 2-9: RR CNAME

```
name   ttl   class   rr   canonical-name
ftp   IN   CNAME   ftp.ejemplo.net.
```

La Tabla 2-7 resume la sintaxis formal para el RR CNAME utilizado en el archivo de zona del Ejemplo 2-9.

Tabla 2-7: RR CNAME

Sintaxis	Descripción
Name	Nombre del host o servidor. Se puede utilizar el formato FQDN o ser un nombre no calificado, provocando la sustitución de $ORIGIN.
TTL	Si no hay ningún valor TTL definido para el RR, será utilizado para la zona el valor por defecto 2d de la directiva $TTL.
Class	Clase del RR.
Canoical-Name	Define que el nombre ftp.ejemplo.com es un alias para el host ftp.ejemplo.net. en un dominio externo o extranjero. En la jerga de DNS, ftp.example.net. se conoce como el nombre canónico que significa simplemente el nombre esperado o real.

El RR CNAME se utiliza a menudo para asignar nombres a servicios existentes en los hosts, por ejemplo, si un host se llama en realidad “fing” pero ejecuta un ftp y un servicio web, los RR CNAME se utilizaran con frecuencia para definir estos servicios, como se muestra en el Ejemplo 2-10:

Ejemplo 2-10: Definición de servicios con CNAME

```
ftp    IN  CNAME fing
www    IN  CNAME fing
fing IN  A     192.168.254.21
```

El RR CNAME tiene algunas limitaciones; en particular, el encadenamiento de registros CNAME es una mala práctica y puede ser origen de errores en las resoluciones DNS.

Ejemplo 2-11: Malas prácticas con el CNAME

```
ns1      IN A     192.168.254.2
mail     IN A     192.168.254.3
Juan     IN CNAME www.ejemplo.com.
www      IN CNAME mail.ejemplo.com.
```

Los registros CNAME no debe utilizarse con recursos NS o MX. Si por ejemplo el servidor de correo y el servidor web se ubicaron en la misma máquina, la definición de estos servicios utilizando el esquema del Ejemplo 2-12 es incorrecta y puede dar error –según el software utilizado–.

Ejemplo 2-12: CNAME y MX

```
        IN MX    mail.ejemplo.com.
mail    IN CNAME www.ejemplo.com.
www     IN A     192.168.254.7
```

El siguiente fragmento resuelve este problema y es válido:

```
IN  MX     mail. ejemplo.com.
```

```
mail  IN  A      192.168.254.7
www   IN  CNAME  mail.ejemplo.com.
```

La regla que define el Ejemplo 2-12 −RFC 1034 [3] sección 3.6.2− recomienda cautela con respecto al uso excesivo de direccionamientos indirectos y dice que, si un nombre aparece en el lado derecho de un RR −como mail.example.com− no debería aparecer en el lado izquierdo del nombre de un RR CNAME.

Otro uso posible del CNAME, es para la corrección ortográfica. Cuando un dominio puede llegar a generar confusiones ortográficas y el operador de zona encuentra que muchas consultas hacia sus dominios fracasan debido a faltas ortográficas, éste puede optar por asociar registros CNAME corrigiendo dichas faltas.

Ejemplo 2-13: CNAME y corrección ortográfica

```
govierno.uy IN TTL CNAME gobierno.uy
```

De esta manera es posible seguir resolviendo una búsqueda aun con faltas ortográficas. Su beneficio puede ser quizás discutible, pero ayuda.

Al utilizar el RR CNAME se tienen dos consecuencias. Primera, CNAME causa que los servidores de nombres realicen más trabajo, dado que tanto el CNAME como el RR CNAME deben ser resueltos por el servidor de nombres. En los servidores de nombres de gran volumen la sobrecarga de trabajo es considerable. Segundo, el RR CNAME y el registro objetivo son devueltos en una respuesta. Cuando se trata de respuestas extensas, esto puede ocasionar que la respuesta exceda el límite de 512 bytes de una transacción DNS UDP, lo que reduce el rendimiento.

2.1.7.1 - Cuando deben ser usado los registros CNAME

En general, el único momento que un CNAME debe ser utilizado, es cuando un host real o canónico se encuentra en un dominio externo.

Ejemplo 2-14: Uso correcto de CNAME

```
; define una IP que se resuelve para ejemplo.com.
      IN    A    192.168.254.7
; alias www.example.com a example.com
www   IN  CNAME example.com.
```

2.1.8 - Registros de Recursos TXT

El RR texto −TXT− históricamente fue utilizo para definir texto genérico que se asocia con un nombre. El texto puede ser cualquier cosa que el usuario desee. Sender Policy Framework −SPF− y las iniciativas de DKIM anti-spam utilizan el RR TXT para transportar información. El TXT también es utilizado por la ISC para proveer su servicio de DLV −DNSSec Lookaside Validation−, ver sección 7.3.1.2 - .

3. Anexo C

3.1 - Registros de Recursos de DNSSec

Aquí resumiremos los nuevos registros de recursos introducidos por DNSSec, información detallada acerca de los mismos se puede encontrar en [13], aquí nos limitaremos a resumir los aspectos más importantes de los mismos para que sirvan de rápida referencia.

DNSSec introduce 4 nuevos tipos de registros.

- DNS Public Key −DNSKEY−
- Resource Record Signature −RRSIG−
- Next Secure −NSEC y NSEC3−
- Delegation Signer −DS−

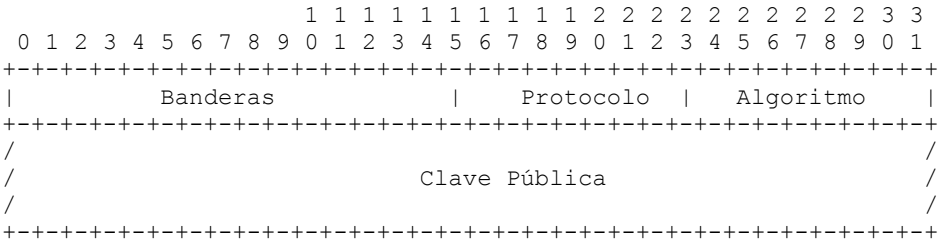
Definiremos el propósito de cada registro, el contenido del RDATA de los RR y su formato de presentación −representación ASCII−. Se especificará el orden canónico de los nombres de dominios y el de los RR dentro de un RRset. Estas dos cuestiones cobran nueva relevancia en DNSSec con respecto a la que tenían en DNS.

3.2 - Registro de Recurso DNSKEY

Cada zona que implemente DNSSec tiene asociadas un par de claves privada y pública, generadas por el administrador de zona. La clave privada debe ser almacenada en lugar seguro y ser protegida por el administrador de la zona. En cambio la clave pública asociada a la zona se publica en el archivo de zona en un registro de recursos DNSKEY.

3.2.1 - Formato del mensaje RDATA DNSKEY

El RDATA del RR DNSKEY consta de dos octetos del campo *Flags*, un octeto de campo *Protocol,* un octeto de campo de *Algorithm* y del campo *Public Key* −clave pública−.



3.2.1.1 - El campo Banderas

El bit 7 del campo Banderas se llama *Zone Key*. Si este bit tiene valor 1, el registro DNSKEY contiene una clave DNSSec de una zona −KSK o ZSK−. Si el bit 7 tiene el valor 0, entonces el registro DNSKEY tiene algún otro tipo de clave pública DNS y no debe ser utilizado para verificar los RRSIGs que cubren los RRsets.

El bits 15 del campo Banderas se denomina Secure Entry Point −Punto de entrada Seguro−, si este tiene valor 1 el registro DNSKEY tiene una clave destinada a ser usada como Secure Entry Point −KSK−. Si de lo contrario el bit está en 0, el registro contendrá una clave ZSK.

Bits 0-6 y 8-14 están reservados, estos bits deben tener el valor 0 en la creación del RR DNSKEY y deberán ser ignorados en recepción.

3.2.1.2 - El campo Protocolo

Este campo debe tener el valor 3, en caso de tener otro valor el RR DNSKEY será tratado como no válido durante la verificación.

3.2.1.3 - El campo Algoritmo

El campo de Algoritmo identifica el algoritmo criptográfico[6] de clave pública y determina el formato del campo de la Clave Pública.

3.2.1.4 - El campo Clave Pública

El campo de la Clave Pública contiene el material de clave pública. El formato depende del algoritmo de la clave.

3.2.2 - Formato de Presentación del RR DNSKEY

El formato de presentación del fragmento RDATA es el siguiente:

[6] En la sección 7.2.2 - se encuentra una lista de los tipos de algoritmo de DNSSec

- El campo *Bandera* debe ser representado como un entero decimal sin signo. Teniendo en cuenta que los posibles valores de las banderas definidas en la actualidad son: 0, 256 y 257.
- El campo *Protocolo* debe ser representado como un entero decimal sin signo con un valor de 3.
- El campo *Algoritmo* debe ser representado como un entero decimal sin signo o como un algoritmo mnemónico.
- El campo *Clave Pública* debe contener la representación codificada en Base 64 de la clave pública. Los espacios en blanco son permitidos dentro del texto Base64.

3.2.3 - Ejemplo RR DNSKEY

El siguiente RR DNSKEY almacena una zone key DNS para example.com.

Ejemplo 3-1: DNSKEY

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                       Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                       kfzj31GajIQKY+5CptLr3buXA10h
                                       WqTkF7H6RfoRqXQeogmMHfpftf6z
                                       Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                       742iU/TpPSEDhm2SNKLijfUppn1U
                                       aNvv4w==  )
```

Los primeros cuatro campos de texto especifican el nombre de dominio, TTL, Clase y el tipo RR −DNSKEY−. El valor 256 indica que el bit Zone Key −bit 7− en el campo Bandera tiene el valor 1. El valor 3 es el valor fijo del campo Protocolo. El valor 5 indica el algoritmo de clave pública, en la sección 7.2.2 - se puede identifica el algoritmo de tipo 5 como RSA/SHA1. El resto del texto es una codificación Base64 de la clave pública.

3.3 - El Registro de Recurso RRSIG

Como ya se vio DNSSec utiliza criptografía de clave pública para firmar y autenticar conjuntos de registros de recursos DNS −RRsets−. Las firmas digitales se almacenan en los registros de recursos RRSIG y se utilizan en el proceso de autenticación DNSSec. Un validador puede utilizar estos RRs RRSIG para autenticar los RRsets de la zona. El RR RRSIG sólo debe ser utilizado para transportar material criptográfico −firma digital−.

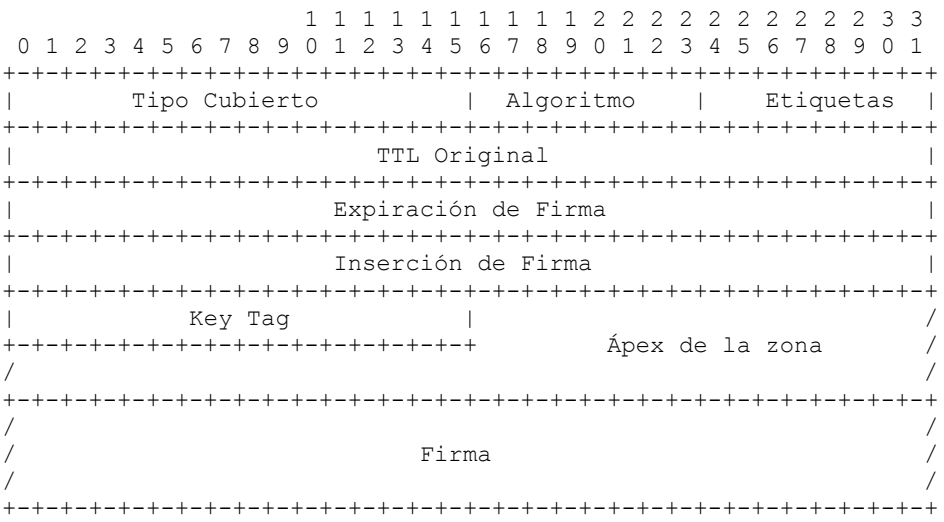
Un registro RRSIG contiene la firma de un RRset con un nombre, una clase y un tipo particular. El RR RRSIG también especifica un intervalo de tiempo durante el cual la firma es válida, el algoritmo utilizado en la generación de la firma, el ápex de la zona,

y la Key Tag para identificar el RR DNSKEY que contiene la clave pública que desencripta la firma.

3.3.1 - Formato del mensaje RDATA RRSIG

El RDATA del RR RRSIG consta de los siguientes campos:

- *Tipo Cubierto* de dos octetos.
- *Algoritmo* de un octeto.
- *Etiquetas* de un octeto.
- *TTL Original* de 4 octetos.
- *Expiración de la firma* de 4 octetos.
- *Inserción de la firma* de 4 octetos.
- *Key-tag* de dos octetos.
- *Ápex de la zona*
- *Firma digital*



3.3.1.1 - El campo Tipo Cubierto

El campo *Tipo Cubierto* identifica el tipo de RRset que cubre este registro RRSIG. Como por ejemplo un tipo A, NSEC o MX entro otros.

3.3.1.2 - El campo Algoritmo

El campo *Algoritmo* identifica el algoritmo criptográfico utilizado para crear la firma. Una lista de tipos de algoritmo de DNSSec se enumera en la Tabla 3-1.

3.3.1.3 - El campo Etiquetas

El campo *Etiquetas* especifica el número original de etiquetas en el nombre de dominio del RR RRSIG. Este campo es importante ya que un validador lo utiliza para determinar si la respuesta fue sintetizada desde de un wildcard o no.

Para validar una firma, el validador necesita el nombre original de dominio, el que se utilizó para crear la firma. Si el nombre original de dominio contiene una etiqueta wildcard –"*"–, el nombre de dominio puede haber sido sintetizado por el servidor durante el proceso de respuesta, en ese caso el validador deberá reconstruir el nombre original de dominio con el fin de validar la firma.

En el valor del campo de Etiqueta no debe contarse la etiqueta nula –la raíz– con la que termina el nombre de dominio o la etiqueta wildcard –si existe–. El valor del campo de Labels debe ser menor o igual al número de etiquetas en el nombre de dominio del RRSIG. Por ejemplo, www.example.com. tiene un valor de campo Etiquetas de 3, y *. example.com. tiene un valor de campo de Etiquetas de 2. La raíz tiene un valor de campo de Etiquetas de 0.

Aunque la etiqueta wildcard no está incluida en el conteo almacenado en el campo Etiquetas de los RR RRSIG, esta es parte del nombre de dominio del RRset cuando la firma es generada o verificada.

3.3.1.4 - El campo TTL Original

El campo *TTL Original* especifica el TTL del RRset cubierto, tal y como aparece en la zona autoritativa.

Este campo es necesario porque un caching resolver debe disminuir el valor TTL de un RRset en cache. Con el fin de validar una firma un validador exige el TTL original.

3.3.1.5 - Campos de Inserción y Expiración de firmas

Estos dos campos especifican un período de validez para la firma. El registro RRSIG no debe ser utilizado para autenticar antes de la fecha indicada en el campo de *Inserción* y no deberá ser utilizado para autenticar después de la fecha que indica el campo *Expiración*.

El formato utilizado en estos campos es un número entero de 32 bits y cuentan los segundos transcurridos desde el 1ro de enero 1970 00:00:00 UTC, ignorando los saltos por segundo. El intervalo más largo que se puede expresar por este formato es de aproximadamente 136 años, pero los valores contenidos en estos campos no pueden hacer referencia a fechas de más de 68 años en el pasado o el futuro.

3.3.1.6 - El campo Key-Tag

Este campo contiene el valor key tag del RR DNSKEY que valida esta firma.

3.3.1.7 - El campo Ápex

El campo *Ápex de la zona* debe contener el nombre de la zona del RRset cubierto. Un remitente no debe utilizar la compresión del nombre DNS en este campo

cuando se transmite un RR RRSIG. Este campo es importante ya que le permite a un servidor validador autenticar a que zona este registro pertenece, evitando así la posibilidad de falsificar registros de otras zonas;

3.3.1.8 - El campo Firma

Este campo contiene la firma criptográfica que cubre el RDATA RRSIG −excepto el propio campo Firma− y el RRset especificado por el nombre de dominio del RRSIG, la clase del RRSIG, y el campo Type Covered del RRSIG. El formato de este campo depende del algoritmo en uso.

La forma de construir la firma digital debe estar bien especificada, tanto el orden de los RR como el formato de los datos son definidos en [13]. Aquí solo describiremos rápidamente la idea básica.

```
signature = sign(RRSIG_RDATA | RR(1) | RR(2)...)
RR(i) = owner | type | class | TTL | RDATA length | RDATA

"|" significa concatenación.
```

Todos los datos deben estar representados en formato “wire”. El RRSIG_RDATA contiene todo el RDATA del registro, menos el campo *Firma.* Resultaría interesante pensar que sucedería si la propia firma se utilizara para generar la firma, pero dejaremos este tema para otro momento. Los RR que conforman el RRset deben estar ordenados de forma canónica, según se describe en la sección 3.7 - de este anexo.

De esta forma queda inequívocamente definido el procedimiento para realizar las firmas, el cual debe ser consistente y bien conocido para que un resolver validador pueda reconstruir el hash firmado y verificarlo contra la firma.

3.3.2 - Formato de Presentación del RR RRSIG

El formato de presentación del fragmento RDATA es el siguiente:

- El campo Tipo Cubierto es representado como un RR tipo mnemónico.
- El valor del campo Algoritmo debe ser representado como un decimal entero sin signo o como un algoritmo mnemónico.
- El valor del campo Etiquetas debe ser representado como un decimal entero sin signo.
- El valor del campo TTL Original debe ser representado como un decimal entero sin signo.
- Los valores de los campos Expiración de Firma e Inserción deben ser representados como un decimal entero sin signo indicando segundos desde el 1ro de enero de 1970 00:00:00 UTC, o en la forma YYYYMMDDHHmmSS en UTC, donde:
    - AAAA es el año −0001-9999−
    - MM es el número del mes −01-12−
    - DD es el día del mes (01-31)

- HH es la hora, en notación de 24 horas −00-23−
- mm son los minutos −00-59−
- SS es el segundo −00-59−

Siempre es posible distinguir entre estos dos formatos porque el YYYYMMDDHHMMSS siempre será exactamente 14 dígitos, mientras que la representación decimal de un número entero de 32 bits sin signo nunca puede ser superior a 10 dígitos.

- El campo Key Tag debe ser representado como decimal entero sin signo.
- El valor del campo Ápex de la zona debe ser representado como un nombre de dominio.
- El campo Firma es representado como una codificación Base64 de la firma. Los espacios en blanco son permitidos en el texto de Base64.

3.3.3 - Ejemplo de RR RRSIG

El siguiente RR RRSIG guarda la firma para el RRset A del dominio host.example.com:

Ejemplo 3-2: RR RRSIG

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                20030220173103 2642 example.com.
                                oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                                PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                                B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rznl8O3t
                                GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                                J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

Los primeros cuatro campos especifican el nombre de dominio, TTL, clase y tipo −RR RRSIG−. La "A" representa el campo Tipo Cubierto. El valor de 5 identifica el algoritmo utilizado −RSA/SHA1− para crear la firma. El valor 3 es el número de Etiquetas en el nombre de dominio original. El valor 86400 en el RDATA RRSIG es el TTL Original del A RRset cubierto. 20030322173103 y 20030220173103 son las fechas de expiración y creación, respectivamente. 2642 es la Key Tag y example.com. es el Ápex de la zona. El resto del texto es una codificación en Base64 de la firma.

La combinación en el RR RRSIG del nombre de dominio, la clase y Tipo Cubierto indican que este RRSIG cubre el RRset A "host.example.com". El valor 3 del Etiquetas indica que ninguna expansión de wildcard se utilizó. El Algoritmo, el Ápex de la zona y el Key Tag indican que esta firma puede ser autenticada con un RR DNSKEY de la zona example.com cuyo algorithm es 5 y cuyo Key Tag es 2642.

3.4 - Registro de Recurso NSEC.

El registro NSEC define el conjunto de tipos de RR para un nombre de dominio. En respuesta a una consulta, si el nombre no existe en el archivo de zona o no existe el

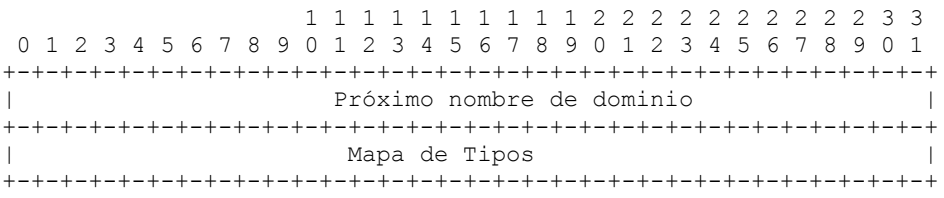
tipo RR para el nombre en cuestión, el registro NSEC es devuelto como prueba autenticada de que el nombre o el tipo de RR no existe.

El conjunto completo de RR NSEC en una zona indica que RRsets autoritativos existen en la zona, formando una cadena de nombres de domino autoritativos para la zona. Esta información se utiliza para proporcionar la autenticación de no existencia de datos en DNS.

Debido a que todos los nombres autoritativos en una zona deben ser parte de la cadena NSEC, los RR NSEC también deben estar presentes en los nombres que contienen un RR CNAME, por lo que esto impone un cambio en las especificaciones tradicionales de DNS[7], que establecen que si existe un CNAME en representación de un nombre sería el único tipo permitido para ese nombre. Pero para una zona firmada debe existir para un mismo nombre un RRSIG y un NSEC, así como existe un registro de recurso CNAME.

3.4.1 - Formato del mensaje RDATA NSEC

El RDATA del RR NSEC se muestra a continuación:



3.4.1.1 - El campo Próximo nombre de dominio

El campo *Próximo nombre de dominio* contiene el siguiente nombre de dominio −en el orden canónico de la zona− que tiene datos autoritativos o que contiene un punto de delegación −RR NS−. El valor de ese campo en el último registro NSEC de la zona, es el nombre del ápex de la zona −el nombre de dominio del RR SOA de la zona−.

Un remitente no debe utilizar la compresión de nombres DNS en este campo cuando se transmite un RR NSEC.

3.4.1.2 - El campo Mapa de Tipos

Este campo indica el tipo de RRset que existen con el nombre de dominio del RR NSEC.

Inclusión de Nombres Wildcard en RDATA NSEC:

[7] Por más información ver RFC 1034

Si un nombre de dominio wildcard aparece en una zona, la etiqueta wildcard –"*"– se trata literalmente como un símbolo y el mismo es tratado como cualquier otro nombre de dominio a los efectos de generar el RR NSEC. Los nombres de dominio wildcard aparecerán en el campo Próximo nombre de dominio sin ningún tipo de expansión de wildcard.

3.4.2 - Formato de presentación del RR NSEC

El formato de presentación del fragmento de RDATA es el siguiente:

- El campo Próximo nombre de dominio es representado como nombre de dominio.
- El campo Mapa de Tipos es representado como una secuencia de RR en su formato mnemónico.

Los siguientes RR NSEC identifican los RRsets asociados con alfa.example.com. e identifican el siguiente nombre autoritativo, después de alfa.example.com.

Ejemplo 3-3: RRsets asociados a alfa.example.com

```
alfa.example.com. 86400 IN NSEC host.example.com. (
                                  A MX RRSIG NSEC TYPE1234 )
```

Los primeros cuatro campos de texto especifican el nombre, TTL, clase y tipo –RR NSEC–. La entrada host.example.com. es el siguiente nombre autoritativo después de alfa.example.com. en el orden canónico. El A, MX, RRSIG, NSEC y mnemónicos TYPE1234 indican que hay RRsets A, MX, RRSIG, NSEC y TYPE1234 asociados con el nombre alfa.example.com.

Suponiendo que el validador puede autenticar este registro NSEC, este podría ser utilizado para demostrar que beta.example.com no existe, o para demostrar que no existe un registro AAAA asociados con alfa.example.com.

3.5 - Registro de Recurso NSEC3

Este tipo de recurso, al igual que NSEC, es utilizado para proveer autenticidad de no existencia en DNSSec. Su funcionamiento es descrito en la sección 0Aquí se especificarán el contenido y formato de este recurso para que sirva de referencia. Información más detallada acerca de este nuevo tipo de recurso se puede encontrar en [20].

A diferencia de su antecesor, el RR NSEC3 utiliza el espacio de nombres hash para proporcionar autenticidad de no existencia. El conjunto completo de estos

registros agrupa todos los RRset que existen para cada nombre de dominio original, y forma una cadena de nombres hash.

3.5.1 - Formato del mensaje RDATA NSEC3

El primer campo, Algoritmo del Hash, indica que tipo de algoritmo fue utilizado para generar el hash de los nombres de dominio. La actual especificación de NSEC3 solo permite el uso de SHA1. El campo Flags es utilizado únicamente para indicar si se está utilizando Opt-Out en la cadena NSEC3.

El número de iteraciones se representa como un entero sin signo de 16 bits, con el bit más significativo primero. El campo Largo Salt indica el largo de la Salt utilizada en la creación de los nombres de dominio hash. Se representa en un solo byte, el tamaño de la salt está limitado por el límite máximo de un nombre de dominio completo, el cual no puede superar los 255 caracteres. Si el largo de la salt es 0, el campo Salt se omite. Si de lo contrario es mayor que cero, el campo Salt debe indicar el string utilizado para la generación del espacio de nombres hash.

El Próximo nombre de dominio hash está codificado en forma binaria, a diferencia del nombre de dominio hash del NSEC3 que esta codificado en Base64. Este campo no debe incluir el nombre de la zona que contiene el registro.

Finalmente, el Mapa de Tipos, al igual que el registro NSEC, codifica los tipos de RRset que el dominio original posee. Se utiliza el mismo método que NSEC en la codificación de este campo.

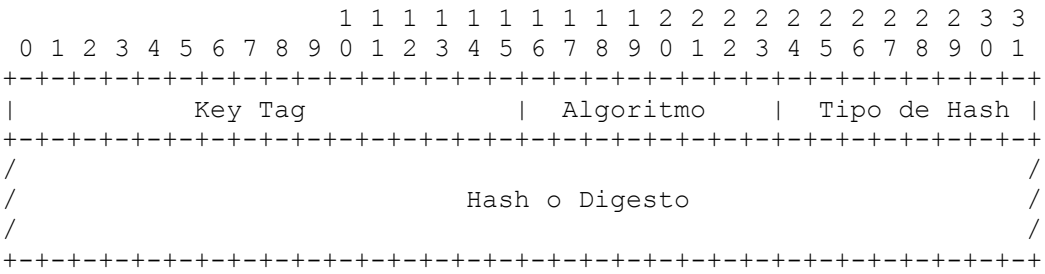
3.6 - Registro de Recurso DS

El Registro de Recurso DS apunta a un RR DNSKEY y es utilizado en el proceso autenticación del RR DNSKEY. Este RR hace referencia a un RR DNSKEY mediante el suministro del Key Tag, el número de algoritmo y un digesto del RR DNSKEY. Con este digesto debería ser suficiente para identificar la clave pública, pero al suministrar el Key Tag y el algoritmo, el proceso de identificación es más eficiente. Mediante la autenticación del registro DS, un resolver puede autenticar el RR DNSKEY al que este registro DS apunta.

El RR DS y sus correspondientes RR DNSKEY tienen el mismo nombre de dominio, pero ellos se almacenan en diferentes lugares. El RR DS sólo aparece en la parte superior −zona padre− de una delegación, mientas que el DNSKEY aparece en el ápex de la zona delegada. Por ejemplo, el RR DS para example.com se almacena en la zona *com* −la zona padre− en lugar de en la zona example.com −la zona hija−. El correspondiente RR DNSKEY se almacena en la zona example.com −la zona hija−. Esto simplifica la administración de la zona DNS y la firma de la zona, pero introduce requisitos especiales de procesamiento de respuesta para el RR DS.

3.6.1 - Formato del mensaje RDATA DS

El RDATA para RR DS consiste en dos octetos de campo Key Tag, un octeto de campo Algorithm, un octeto de campo Digest Type y un campo Digest.



3.6.1.1 - El campo Key Tag

Este campo lista el key tag del RR DNSKEY a el cual el registro DS hace referencia o apunta. El Key Tag utilizado por el RR DS es idéntico a la Key Tag utilizado por los RR RRSIG.

3.6.1.2 - El campo Algoritmo

El campo *Algoritmo* lista el número de algoritmo del RR DNSKEY a el cual el registro DS hace referencia o apunta. El número de algoritmo utilizado por el RR DS es idéntico al número de algoritmo utilizado por los RR RRSIG y DNSKEY.

3.6.1.3 - El campo Tipo de Hash

El RR DS hace referencia a un RR DNSKEY cuando incluye un digesto de ese RR DNSKEY. Este campo identifica el algoritmo utilizado para la construcción del digesto.

3.6.1.4 - El campo Digesto

Como ya se menciono el registro DS hace referencia a un RR DNSKEY mediante la inclusión de un digesto del RR DNSKEY. Este digesto se calcula mediante la concatenación de la forma canónica del nombre completo de dominio del RR DNSKEY con la DNSKEY RDATA y luego aplicar el algoritmo de digesto.

```
digest = digest_algorithm( DNSKEY owner name | DNSKEY RDATA);

"|" denota la concatenación

DNSKEY RDATA = Flags | Protocol | Algorithm | Public Key.
```

El tamaño del Digest puede variar dependiendo del algoritmo digesto y el tamaño del RR DNSKEY.

3.6.2 - Procesamiento de RR DS al validar las respuestas.

El RR DS une la cadena de autenticación a través de fronteras de zonas, por lo que el RR DS requiere un especial cuidado en su procesamiento. El RR DNSKEY al que apunta el RR DS debe ser una clave de zona de DNSSec, entonces:

- Las banderas de los DNSKEY RR deben tener seteado el bit 7.
- Si las banderas DNSKEY no indican una clave de zona DNSSec, el RR DS –y el RR DNSKEY al que hace referencia–, no deberán ser utilizados en el proceso de validación.

3.6.3 - Formato de presentación del RR DS.

El formato de presentación de la parte RDATA es el siguiente:

- El campo Key Tag debe ser representado como un entero decimal sin signo.
- El campo de Algorithm debe ser representado como un entero decimal sin signo o como un algoritmo mnemónico.
- El campo Digest Type debe ser representado como un entero decimal sin signo.
- El Digesto debe ser representado como una secuencia de dígitos hexadecimales entre mayúsculas y minúsculas. Los espacios en blanco son permitidos en el texto hexadecimal.

3.6.4 - Ejemplo RR DS.

El siguiente ejemplo muestra un RR DNSKEY y sus correspondientes RR DS.

Ejemplo 3-4:RR DNSKEY y sus correspondientes RR DS

```
dskey.example.com. 86400 IN DNSKEY 256 3 5 ( AQOeiiR0GOMYkDshWoSKz9Xz
                                            fwJr1AYtsmx3TGkJaNXVbfi/
                                            2pHm822aJ5iI9BMzNXxeYCmZ
                                            DRD99WYwYqUSdjMmmAphXdvx
                                            egXd/M5+X7OrzKBaMbCVdFLU
                                            Uh6DhweJBjEVv5f2wwjM9Xzc
                                            nOf+EPbtG9DMBmADjFDc2w/r
                                            ljwvFw==
                                            ) ;  key id = 60485
dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A
                                            98631FAD1A292118)
```

Los primeros cuatro campos de texto especifica el nombre, TTL, clase y tipo −RR DS−. El valor 60485 es la Key Tag para el correspondiente RR DNSKEY dskey.example.com., y el valor de 5 indica el algoritmo utilizado por este RR DNSKEY dskey.example.com. El valor 1 es el algoritmo utilizado para la construcción del digesto y el resto del texto RDATA es el digesto en formato hexadecimal.

3.7 - Forma Canónica y Orden de los Registros de Recursos

En esta sección se definirá la forma canónica de los registros de recursos, un orden canónico de los nombres DNS y un orden canónico de los registros de recursos dentro de un RRset. Un orden canónico de nombres se requiere para construir la cadena de nombre NSEC. La forma canónica de los RR y el orden canónico dentro de un RRset se requieren para construir y verificar los RR RRSIG.

3.7.1 - Orden canónico de nombres DNS

Con el fin de obtener seguridad en el DNS los nombres de dominios están ordenados de forma de tratar de manera individual cada cadena de octeto sin signo. Las letras en mayúsculas US-ASCII son tratadas como si fueran letras minúsculas US-ASCII.

Para determinar el orden canónico de un conjunto de nombres DNS, se empieza por ordenar los nombres de acuerdo a sus etiquetas más significativas −la de más a la derecha−. Para los nombres en los que estas etiquetas son idénticas, se sigue ordenando de acuerdo a su siguiente etiqueta más a la izquierda −menos significativa en la jerarquía DNS−.

Por ejemplo, los siguientes nombres están ordenados en el orden canónico de nombres DNS. La etiqueta más importante es "ejemplo". En este nivel, se ordena en primer lugar "ejemplo", seguido de los nombres que terminan en "a.example" y luego por los nombres que terminan "z.example". Los nombres dentro de cada nivel se clasifican de la misma manera.

Ejemplo 3-5: Orden canónico

```
example
a.example
yljkjljk.a.example
Z.a.example
zABC.a.EXAMPLE
z.example
\001.z.example
*.z.example
\200.z.example
```

3.7.2 - Forma Canónica de los RR.

El orden canónico de un RR se define según su formato “wire”, donde:

- Cada nombre de dominio en el RR está totalmente expandido y calificado −sin compresión del nombre DNS−.
- Todas las letras mayúsculas US-ASCII en el nombre de dominio son sustituidas por sus respondientes letras minúsculas US-ASCII.
- Si el tipo de RR es NS, MD, MF, CNAME, SOA, MB, MG, MR, PTR, HINFO, MINFO, MX, HINFO, RP, AFSDB, RT, SIG, PX, NXT, NAPTR, KX, SRV, DNAME, A6, RRSIG o NSEC, todas las letras mayúsculas US-ASCII de los nombres de DNS contenidos dentro del RDATA son sustituidas por su correspondiente letras en minúsculas US-ASCII.
- Si el nombre de dominio del RR es un nombre wildcard, el nombre de dominio está en su forma no expandida, incluyendo la etiqueta "*" −no sustitución de wildcard−.
- El TTL del RR es puesto en su valor original tal y como aparece en la zona autoritativa de la que proviene o el campo Original TTL del RR RRSIG cubierto.

3.7.3 - Orden Canónico RR dentro de un RRset

Un RRset no puede contener registros duplicados −RR múltiples con el mismo nombre de dominio, clase, tipo y RDATA−. Por lo tanto, si una aplicación detecta RRs duplicados al poner el RRset en forma canónica, se deberá tratar esto como un error de protocolo. Una implementación puede optar por manejar este error eliminando todos menos uno de los duplicados RRs para el cálculo de la forma canónica del RRset.

3.8 - Registros y Seguridad.

Aparte de los elementos descritos a continuación, los registros de recursos por ellos mismos no aportan seguridad al protocolo.

Un registro DS identifica un RR DNSKEY existente mediante el uso de un digesto, el tipo de algoritmo de clave y un key tag, por lo que la probabilidad que un atacante construya un DNSKEY que coincida con estos campos dado un registro DS es muy baja.

El Key tag es utilizado para ayudar a seleccionar eficientemente los registro de recursos DNSKEY, pero no identifica exclusivamente un único registro de recursos DNSKEY. Es posible que dos RR DNSKEY distintos tengan el mismo nombre de dominio, el mismo tipo de algoritmo y el mismo key tag. Una aplicación que utilice solamente la key tag para seleccionar un RR DNSKEY puede llegar a seleccionar la clave pública equivocada en ciertas circunstancias.

3.9 - Algoritmos DNS y tipos de Digestos.

Las extensiones de seguridad de DNS están diseñadas para ser independientes de los algoritmos criptográficos subyacentes. Los registros de recursos DNSKEY, RRSIG y DS, todos utilizan un Número de Algoritmo DNSSec para identificar el algoritmo criptográfico utilizado por el registro de recursos. El registro de recursos DS también especifica un Número de Algoritmo Digesto para identificar el algoritmo digesto utilizado para construir el registro DS. El Algoritmo y Digesto definidos actualmente se enumeran a continuación. Algoritmos adicionales o Tipos de Digesto puede ser añadido con los avances de la criptografía.

Un resolver DNSSec o servidor de nombres security-aware debe implementar todos los algoritmos obligatoriamente.

3.9.1 - Tipos de algoritmos DNSSec.

Los RR DNSKEY, RRSIG y DS utiliza un número de 8 bits para identificar el algoritmo de seguridad que utiliza. Estos valores se almacenan en el campo "Número Algoritmo" en el RDATA del registro de recursos. Algunos de los algoritmos se pueden utilizar sólo para la firma de la zona −DNSSec−, algunos sólo por mecanismos de seguridad en las transacciones −SIG (0) y TSIG− y algunos para ambos. Los utilizables para la firma de la zona pueden aparecer en DNSKEY, RRSIG, y RR DS.

Tabla 3-1: Algoritmo de claves DNSSec

Valores	Algoritmo
0	Reservado
1	RSA-MD5
2	Diffie-Hellman (RFC 2539)
3	DSA/SHA-1—Opcional (RFC 3755, 2536)
4	Elliptic curve—Todavía no estandarizado
5	RSA/SHA-1—Obligatorio (RFC 3755, 3110)
6	DSA-NSEC3-SHA1 (RFC 5155)
7	RSASHA1-NSEC3-SHA1 (RFC 5155)
8	RSA/SHA-256 (RFC 5702)
9	Sin Asignar
10	RSA/SHA512 (RFC 5702)
11	Sin Asignar
12	GOST R 34.10-2001 (RFC 5933)
13-122	Sin Asignar
123-251	Reservado
252-254	Sin Asignar
255	Reservado

3.9.2 - Tipos de Digestos DNSSec

El campo *Digest Type* en el tipo RR DS identifica el algoritmo digesto criptográfico usado por los RR. La siguiente tabla lista los tipos de algoritmos digestos definidos en la actualidad.

Tabla 3-2:Tipos de digestos

Valores	Algoritmo
0	Reservado
1	SHA-1
2-255	Unassigned

3.10 - Consideraciones del Key Tag.

El campo *Key Tag* en los RR RRSIG y DS proporciona un mecanismo eficiente de selección de una clave pública. En la mayoría de los casos, la combinación del nombre de dominio, el algoritmo y el key tag pueden de manera eficiente identificar un registro DNSKEY. El campo Key Tag en el RR RRSIG y en el DS puede ser utilizado para ayudar a seleccionar eficientemente el correspondiente RR DNSKEY, cuando más de un candidato RR DNSKEY está disponible.

Sin embargo, es fundamental tener en cuenta que el key tag no es un identificador único. En teoría, es posible que dos RR DNSKEY distintos tengan el mismo nombre de dominio, el mismo algoritmo y el mismo key tag. El key tag es utilizado para limitar las posibles claves candidatas, pero este no identifica de forma única un registro DNSKEY. Las implementaciones no deben asumir que el key tag es el único identificador de un DNSKEY RR.

El key tag es el mismo para todos los tipos de algoritmo DNSKEY excepto para el algoritmo 1. El algoritmo key tag es la suma del formato del mensaje del RDATA DNSKEY divido en dos grupos de octetos. En primer lugar, el RDATA −en formato de mensaje− es tratado como una serie de dos grupos de octetos. Estos grupos después son sumados juntos ignorando cualquier bit acarreado.

4. Anexo D

4.1 - Tipos de servidores DNS

4.1.1 - Comportamiento de un servidor esclavo DNS

Los servidores esclavos responderán con autoridad a preguntas del dominio mientras mantengan validos los registros de la zona, sin ningún tipo de diferencia para el cliente. Esta característica permite al maestro de la zona estar oculto del acceso público –configuración maestro oculto– otorgándole al sistema un extra de seguridad. Para ilustrar por qué tal estrategia puede ser útil, consideremos los siguientes escenarios: si un archivo de la zona esclavo se ve corrompido por un ataque malévolo, rápidamente puede ser restaurado por una transferencia de zona desde el maestro. Si el archivo de zona maestra se corrompe de manera similar, el archivo de zona debería ser restaurado a partir de los medios de copia de seguridad, que podría tomar algún tiempo. Una forma de prevenir este problema es simplemente evitándolo, haciendo que el maestro no pueda ser accedido de forma pública y por lo tanto no estar sirviendo a la zona de forma activa –respondiendo consultas–. Éste es únicamente visible por los esclavos, y no aparecería en ningún RR NS para la zona. Cada servidor de nombre, maestro o esclavo, que el usuario desee hacer visible debe ser definidos usando un RR NS en la zona.

La Figura 4.1 ilustra a un maestro típico y una única configuración esclavo y la Figura 4.2 ilustra a un servidor esclavo cuando se utiliza un maestro oculto.

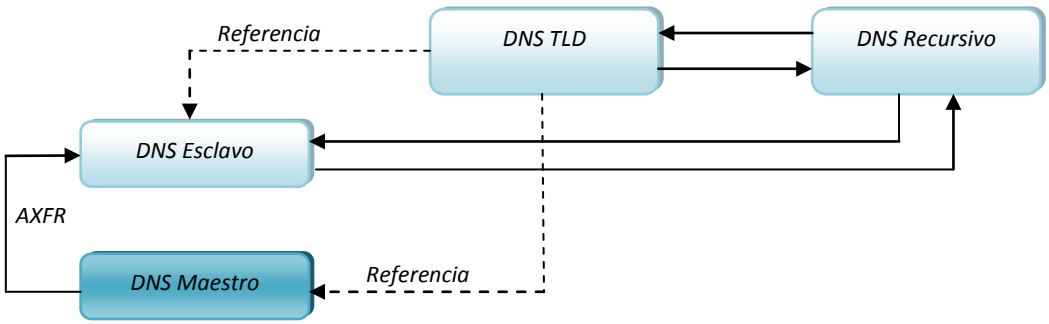


Figura 4.1: Configuración típica maestro-esclavo

4.1.2 - Esclavo vs. cache

Un servidor esclavo obtiene todos los datos de zona a la cual sirve a través de las operaciones de transferencia de zona. Este proceso no debe ser confundido con una memoria cache. El servidor esclavo usa los valores de refresh y expiry del RR SOA para solicitar actualizaciones y dar de baja los datos de su zona. Un cache por otra parte, contiene los RR individuales obtenidos como respuesta a una consulta específica, los cuales cuando su TTL es alcanzado caducan del cache. Además, un servidor esclavo siempre responde con autoridad a las solicitudes de información acerca de su zona. El cache sólo responderá con datos de zona de autoridad la primera vez que obtiene los datos del maestro o del esclavo. A partir de entonces, cuando la lectura es desde su cache, los datos no contendrán el flag AA indicando autoridad. Solo

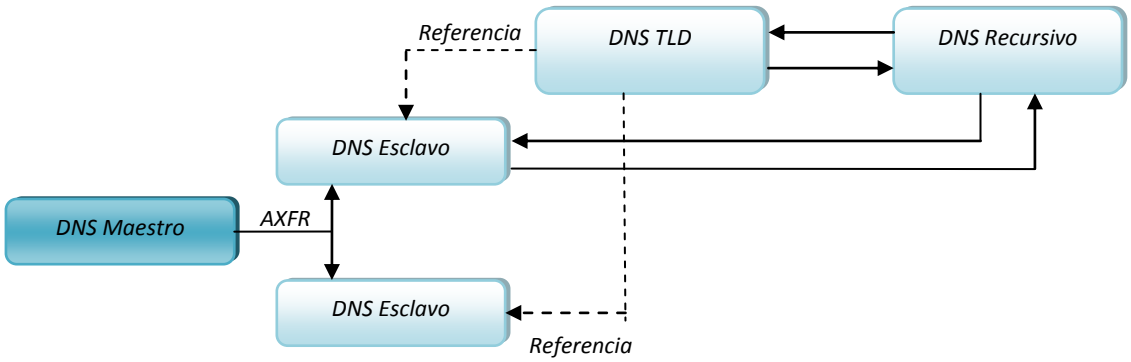


Figura 4.2: Configuración Maestro Oculto

un servidor maestro de zona o un esclavo autorizado pueden aparecer en cualquier RR NS de zona, ya que sólo estos tipos de servidores pueden responder con autoridad para la zona.

4.1.3 - Propagación de cambios usando NOTIFY

El esclavo periódicamente –según el valor refresh del RR SOA– preguntará al maestro de la zona por cambios en la zona. El parámetro refresh, que típicamente puede ser 12 horas o más, determina el tiempo de propagación de los cambios de zona. Si el comportamiento NOTIFY está habilitado para la zona maestra –función por defecto en el BIND– entonces, cada vez que se carga o se recarga la zona un mensaje NOTIFY se envía a todos los servidores esclavos definidos en el RR NS del archivo de zona. Al recibir un mensaje NOTIFY el esclavo solicitará una copia del RR SOA de la zona. Si el número de serie actual de los datos de zona es menor que el número de serie del nuevo RR SOA, el esclavo inicia una transferencia de zona para actualizar completamente sus datos de zona.

El mensaje NOTIFY –y sus posteriores operaciones de transferencia de zona– se presenta como una amenaza para la seguridad. Para reducir al mínimo esta amenaza, el comportamiento por defecto de BIND es sólo aceptar los mensajes NOTIFY del

maestro de zona –servidores de nombres listados en la declaración *masters* en las cláusulas de la zona–. Otras fuentes admisibles de NOTIFY se pueden definir mediante la declaración allow-notify en el archivo named.conf.

4.1.4 - Servidores de nombres Forwarding –Proxy–

Un servidor DNS de forwarding −proxy o remoto− es aquel que reenvía todas las consultas DNS a otro y guarda los resultados en el cache. Un servidor de nombres de forwarding puede ser útil en los siguientes casos, cuando el acceso a una red externa es lento, caro o está fuertemente congestionado.

- El servidor de nombres al que las consultas son reenviadas suministrará soporte de consulta recursiva, lo que resulta en una sola transacción DNS. Si el servidor de nombres local fuera un servidor de caching-only y no reenviara las consultas, las múltiples transacciones que se producen incrementaría la carga y los retrasos en la red.
- El DNS forwarding local guardará en memoria los resultados y por lo tanto, proporcionará respuestas más rápidas a la información a la que se accede más frecuentemente, minimizando el innecesario −y potencialmente inseguro− tráfico externo.

El re envió también se puede utilizar como parte de un configuración stealth −o división− del servidor, que se describe en la siguiente sección. En la Figura 4.3 se ilustra el uso del DNS forwarding.

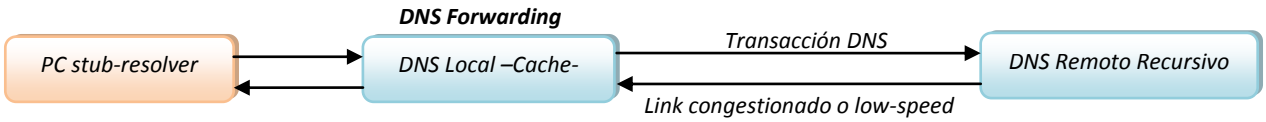


Figura 4.3: DNS Forwarding

BIND permite la configuración forwarding utilizando las declaraciones *forward* y *forwarders* ya sea a nivel global o para cada zona. Ambas configuraciones se muestran en los ejemplos siguientes.

Este fragmento de named.conf provoca un reenvío global de todas las consultas recibidas por el servidor de nombres:

Ejemplo 4-1: Cláusula Global

```
// Clausula de opciones globales
// Forwarders
options {
            forwarders {10.0.0.1; 10.0.0.2;};
            forward only;
};
```

Definiendo la sentencia *forwarders* dentro de la *cláusula de opciones*, ésta se aplica a toda la configuración del servidor a menos que exista otra sentencia en una cláusula de la zona posterior que la sobrescriba. Los *forwarders* y *forward* se utilizan siempre conjuntamente unos con otro.

4.1.5 - Servidor de nombres Stealth –DMZ or Split–

Un servidor stealth se define como un servidor de nombres que no aparece públicamente visible en ninguna RR NS para el dominio. Servidores stealth se utilizan en las configuraciones que a veces se llaman *zonas* desmilitarizadas –DMZ– o servidores de división y se definen según las siguientes características:

- La organización necesitan exponer servidores DNS para proporcionar acceso a los servicios públicos, algunos sitios web, correo electrónico, sitios FTP, etc.
- La organización no desea que el mundo vea ninguno de sus servidores internos, ya sea a través de consultas DNS o a causa de un mal funcionamiento.

Una arquitectura de servidor stealth o división se ilustra en la Figura 4.4.

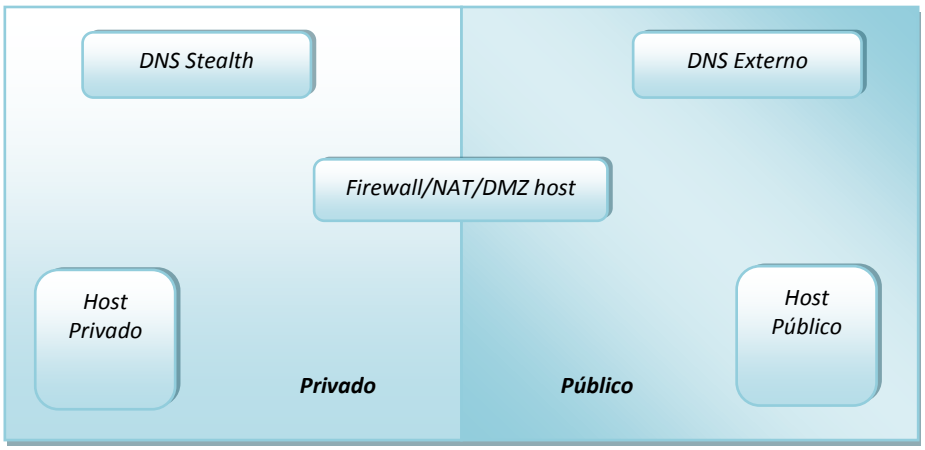


Figura 4.4: DNS Stealth

Los servidores externos o públicos están configurados para proporcionar solamente respuestas con autoridad y no servicios de caching. En este caso, el almacenamiento en cache constituye un desperdicio en términos de rendimiento y una posible fuente de contaminación o corrupción, las cuales llegan a comprometer el sistema. El archivo de zona utilizado por los servidores públicos, es un subconjunto de datos del archivo de zona y contendrá sólo aquellos sistemas o servicios que el usuario necesita para hacerse externamente visible. Por ejemplo, un RR SOA, RR NS para servidores –no stealth– de nombres públicos, RR MX para los servidores de correo y RR A. La transferencia de zona se permite entre los servidores de nombres públicos cuando sea necesario, pero no debe transferir o aceptar transferencias desde el

servidor stealth. Esta clara separación entre la esfera privada y pública de la red es necesaria porque si el servidor de nombres público se ve comprometida, entonces la simple inspección del archivo named.conf o los archivos de zona no debe proporcionar información que describa ninguna parte oculta de la red. En named.conf de BIND 9 las directrices como *masters*, allow-notify, allow-transfer y otros, si están presentes, proporcionan información que permite a un atacante penetrar el velo de la privacidad.

4.1.5.1 - Servidores Stealth y la Cláusula View

BIND 9 proporciona una cláusula view que se pueden utilizar para proporcionar funcionalidades similares a un solo servidor, pero esto no resuelve el problema cuando el servidor se ve comprometido, revelando datos adicionales acerca de la organización a través de una simple inspección del named.conf y los archivos de zona. La declaración *view* de BIND puede utilizarse para aumentar la funcionalidad de las partes públicas y privadas de una configuración de stealth.

5. Anexo E

5.1 - Otras operaciones del DNS

5.1.1 - Consultas Inversas

Una consulta inversa asigna un RR a un dominio. Una consulta inversa podría ser por ejemplo "¿Cuál es el nombre de dominio para este registro MX?". El soporte de la consulta inversa siempre se definió como un servicio opcional dentro de las especificaciones del DNS, y se permitió a los servidores de nombres devolver una respuesta de "No implementado" –NOTIMP-. En consecuencia, las consultas inversas no se han utilizado ampliamente y quedaron obsoletas por la RFC 3425 [48].

5.1.2 - DNS Reverse Mapping

Dado un nombre de dominio, una consulta normal de DNS trata de determinar su dirección IP. A veces, sin embargo, resulta útil poder determinar el nombre del host dada su dirección IP. Esto puede ser necesario para fines de diagnóstico y se utiliza también por razones de seguridad, para rastrear un hacker o un spammer. De hecho los sistemas de correo más modernos utilizan revese mapping para proporcionar la autenticación simple, mediante el uso de políticas de búsqueda de DNS -por ejemplo, IP-a-nombre y nombre-a-IP- para confirmar que la dirección IP especificada representa el host indicado. Con el fin de realizar la asignación inversa usando consultas recursivas e iterativas normales los diseñadores del DNS definieron un nombre de dominio especial –reservado– llamado IN-ADDR.ARPA.

5.1.2.1 - IN-ADDR.ARPA Reverse-Mapping Domain

La estructura del nombre de dominio normal es jerárquica a partir de la raíz. Un nombre de dominio está escrito de izquierda a derecha, pero la estructura jerárquica se escribe de derecha a izquierda. Por ejemplo www.example.com. el nodo más alto en la jerarquía de DNS −o árbol− es la raíz, que se define normalmente por el punto –aunque suele omitirse al colocar el nombre−. Esto es seguido por *com*, el dominio de primer nivel −TLD−. Siguiente –inferior− es *.ejemplo*, el dominio de segundo nivel, y finalmente el más bajo es www, que es el nombre de host y se define siempre en un archivo de zona. Para habilitar una dirección IPv4 para ser utilizada en una operación de consulta normal debe ser convertido en un nombre de dominio, tal como se describe a continuación. Una dirección IPv4 se escribe como sigue:

Ejemplo 5-1: Dirección IP

```
192.168.254.17
```

Esta dirección IPv4 define una dirección de host de 17 en un rango de direcciones de Clase C 192.168.254.x. En este caso, la parte más importante −el nodo superior− está a la izquierda −192−, no a la derecha. Esto es un poco torpe y haría imposible la construcción de una estructura de árbol donde pueda realizarse una búsqueda eficiente. La solución es elegantemente simple; para crear el nombre de dominio, invertir el orden de la dirección y la construcción de la jerarquía, bajo el nombre de dominio especial IN-ADDR.ARPA -el SLD es IN-ADDR, el TLD es ARPA-.

Finalmente, la última parte de la dirección IPv4 −17− es la dirección de host. El resultado de la manipulación anterior es el siguiente:

Ejemplo 5-2: IN-ADDR.ARPA

```
IPv4 address = 192.168.254.17
Clase base C = 192.168.254; omite la dirección del host = 17
Clase C base invertida = 254.168.192

Añadido a IN-ADDR.ARPA dominio = 254.168.192.IN-ADDR.ARPA.
```

La organización del dominio IN-ADDR.ARPA se muestra en la Figura 5.1.

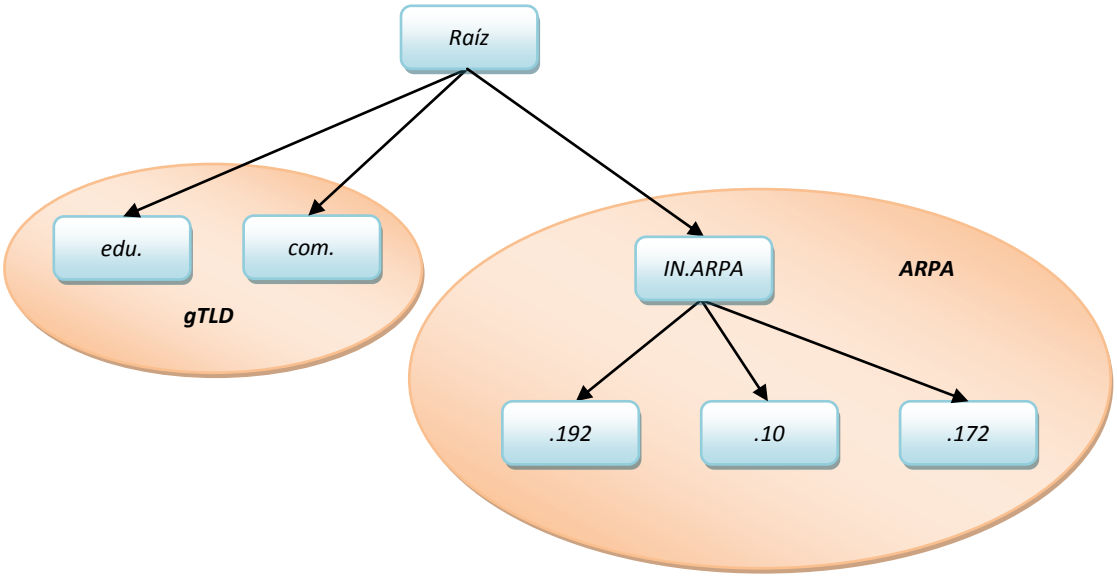


Figura 5.1: Zona ARPA

Por último, se construye un archivo de la zona describiendo todos los hosts en la zona mapeada inversamente utilizando el RR PTR.

5.1.2.2 - The PTR Resource Record

Los RR PTR están estandarizados en la RFC 1035 [4] y asignan una dirección IPv4 a un host en particular en el dominio o zona, en comparación con un RR A que asigna un nombre a una dirección IPv4 o un RR AAAA que asigna un nombre a una dirección IPv6.

Una dirección IPv4 se puede mapear a uno o más nombres de host utilizando los RR PTR. Donde varios de los RR A o CNAME se pueden utilizar para definir la misma dirección IPv4, todos pueden ser mapeados a la misma IPv4 −o IPv6− en el archivo de zona IN-ADDR.ARPA. Un conjunto de RRs PTR mapeando la misma dirección IP a diferentes hosts se denomina RRset PTR.

5.1.2.3 - Reverse-Map Queries

Reverse-map queries utilizan las consultas normales recursivas o iterativas bajo el dominio especial IN-ADDR.ARPA. El dominio ARPA está estructurado jerárquicamente con ICANN/IANA en la raíz y es administrado conjuntamente por la ICANN/IANA y el IETF/IAB −RFC 3172 [49]−. A diferencia de los dominios hacia adelante, que utilizan los servidores gTLD o ccTLD como el siguiente nivel de delegación, las direcciones IPv4 son delegadas a través de los Registros Regionales de Internet −RIR−, que se muestran en Tabla 5-1.

Tabla 5-1: Registros Regionales de Internet –RIR-

Nombre de RIR	Cobertura	Web
APNIC	Asia y el Pacífico	www.apnic.net
ARIN	Norte de América y parte del Caribe	www.arin.net
LACNIC	Sur de América y parte del Caribe	www.lacnic.net
RIPE NCC	Europa, Medio Oriente y parte de Asia	www.ripe.net
AFRINIC	África	www.afrinic.net

Los RIR operan bajo los procedimientos definidos en la RFC 2050 [50]. Las direcciones IPv4 se asignan en bloques de red por los RIR ya sea a un Registro de Internet Local –LIR–, por lo general un ISP, o de un Registro Nacional de Internet –NIR– que a su vez se asignan a un LIR. A cada nivel de registro de Internet se le ha delegado la responsabilidad de revertir el mapping de las direcciones que se le han asignado. La LIR puede delegar la responsabilidad de invertir la asignación para el usuario final si se utilizan direcciones IPv4 estáticas. Sin embargo, la organización de la correspondencia inversa se basa en cada valor separado por puntos en una dirección IP. Si la última parte de la dirección IPv4 asignada a un usuario final es una subred -menos de 256 direcciones-, entonces surge un problema ya que cualquier entidad, un nombre de dominio o de un bloque de direcciones, en la jerarquía de dominio se puede delegar una vez y sólo una vez. En el caso de una subred el bloque de red mismo tendría que ser delegado a cada usuario de subred. Para ilustrar este punto supongamos que el bloque de red 192.168.254.0 es que se asignarán a cuatro usuarios cada uno de los cuales tendrá 64 direcciones –una subred de 64 direcciones–. Que se asignarán como se muestra en la notación de prefijo IP:

Ejemplo 5-3: División de bloque de red.

```
Primer usuario- 192.168.254.0/26 (máscara de 255.255.255.192)
Segundo usuario - 192.168.254.64/26
Tercer - 192.168.254.128/26
Cuarto - 192.168.254.192/26
```

Cuando el bloque de red para este grupo se asigna inversamente, la parte del equipo se omite dando 192.168.254 y luego invertido en el dominio IN-ADDR.ARPA, lo siguiente:

Ejemplo 5-4: Invertido en el dominio IN-ADDR.ARPA

```
254.168.192.IN-ADDDR.ARPA
```

Cada uno de los cuatro usuarios requiere la delegación de este dominio con el fin de proporcionar la asignación inversa de su propio rango de direcciones asignado. Esto incumple el principio de una sola delegación. Para superar esta limitación, la construcción de reverse-mapping para la delegación de subredes utiliza una construcción especializada de reverse-map, básicamente crea un espacio de nombres adicionales.

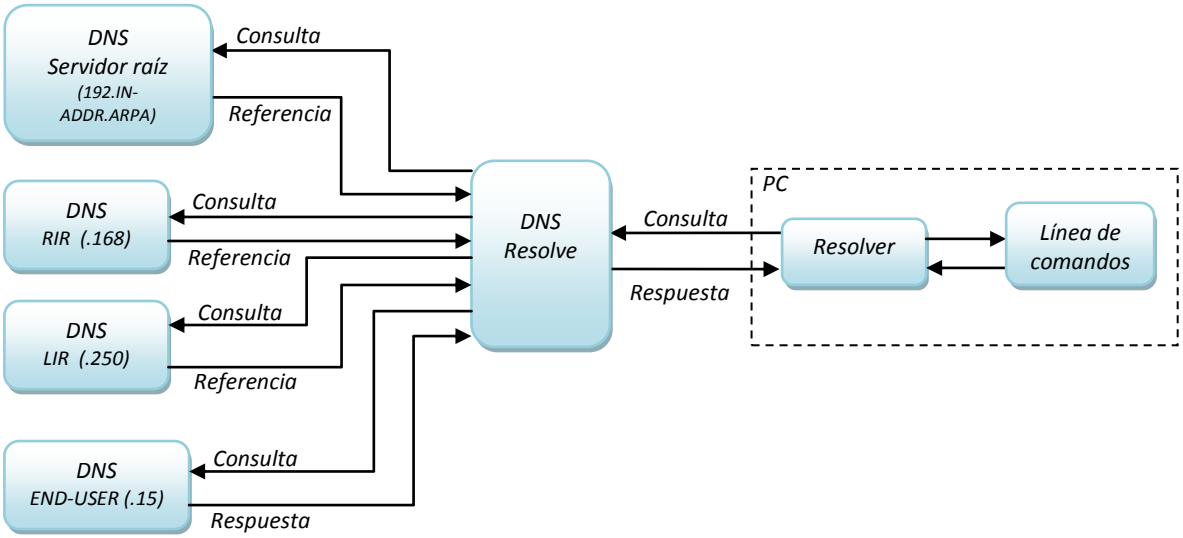


Figura 5.2: Consulta recursiva por 15.250.168.IN-ADDR.ARPA

En la Figura 5.2 se puede observar una consulta recursiva la cual intenta contando con la IP resolver el nombre de dominio:

5.1.3 - Transferencia completa de zona –AXFR–

Cuando hablamos de zone transfer o transferencia de zona nos referimos a la forma en que un servidor esclavo actualiza el contenido de su archivo de zona desde el servidor maestro. Este proceso permite a un servidor de nombres secundario mantener su archivo de zona sincronizado con su servidor de nombres primario.

Las especificaciones originales de DNS –RFC 1034 [3] y RFC 1035 [4]– prevén que el servidor de nombres esclavo consulte al servidor de nombres maestro para la zona. El tiempo entre sondeo se determina por el valor de refresh del RR de dominio SOA. En un archivo de zona típico este valor será de 12 horas o más. El proceso de consulta DNS es llevado a cabo por el servidor de nombres esclavo, consultando al maestro solicitando el RR SOA. Si el número de serie del RR SOA es mayor que el actual que mantiene el servidor de nombres esclavo, una transferencia de zona completa –AXFR– es solicitada por el DNS esclavo. Esta es la razón por la cual es vital la actualización del número de serie SOA cada vez que algo cambia en cualquiera de los registros de la zona.

5.1.4 - Transferencia de zona incremental –IXFR–

Transferir archivos de zona muy grandes puede tardar mucho tiempo así como también malgastar ancho de banda y otros recursos. Esto es especialmente infructuoso si sólo un único registro se ha cambiado. La RFC 1995 [51] introdujo la transferencia de zona incremental -IXFR-, que permite que el servidor de nombres esclavo y el servidor de nombres maestro transfieran sólo aquellos registros que han cambiado. El servidor de nombres esclavo envía una consulta por el SOA al maestro en cada intervalo de

actualización. Si el número de serie del RR SOA es mayor que el que actualmente está almacenado por el esclavo, el servidor solicita una transferencia de zona e indica si es o no es capaz de aceptar un IXFR. Si ambos servidores maestro y esclavo soportan la función un IXFR se lleva a cabo, de lo contrario, un AXFR se lleva a cabo. IXFRs usa TCP en el puerto 53.

IXFRs sólo afecta el volumen de datos que se transfieren no tienen impacto en el tiempo que tardan en propagarse los cambios del archivo de zona.

5.1.5 - Notify

RFC 1912 [46] recomienda un intervalo de 2 a 12 horas o incluso más en el intervalo de actualización para el RR SOA. Esto significa que los cambios en la zona maestra pueden no ser visibles para la zona esclava hasta 12 horas o más luego de realizados, lo que sea establecido en este valor. Dado que este retraso no es aceptable para lo que es Internet hoy en día en la RFC 1996 [42] introdujo un plan para que un servidor de la zona de nombres autoritativo −maestro o esclavo− envíe un mensaje de N*otify* a la zona de los servidores de nombre −definido por el RR NS para la zona− cada vez que la zona está cargada o actualizada. Este mensaje indica que un cambio puede haber ocurrido en los registros del dominio. El servidor de nombres en la recepción del mensaje NOTIFY solicitará el RR SOA de la zona principal y si el número de serie es mayor que el que actualmente está almacenado realizará una transferencia de zona utilizando un AXFR o un IXFR. NOTIFY puede reducir considerablemente el tiempo necesario para propagar los cambios de zona a los servidores.

5.1.6 - Dynamic Update

El método clásico para la actualización de los RR de zona es editar manualmente el archivo de zona y luego detener e iniciar el servidor de nombres para leer los archivos de zona y propagar los cambios. Esto puede ser inaceptable cuando el volumen y la frecuencia de los cambios alcanzan cierto nivel, también nos encontramos con aplicaciones que requieren actualizaciones automáticas del archivo de zona DNS. Algunas de estas aplicaciones incluyen Certificate Authority Servers −CA−, servidores Dynamic Host Configuration Protocol −DHCP− e Internet multicast address servers. Por lo que muchos de los usuarios más grandes de DNS buscaron un método para cambiar rápidamente los registros de la zona, mientras que el servidor de nombres sigue respondiendo a las consultas de los usuarios. Hay dos enfoques arquitectónicos para resolver este problema:

- Permitir en un tiempo de ejecución la actualización de los registros de zona por medio de una fuente externa o aplicación.
- Directamente alimentar los RRs de la zona a partir de una base de datos, que puede ser actualizada dinámicamente.

En la RFC 2136 [52] se toma un primer enfoque y define un proceso llamado DNS dinámico –DDNS-, mediante el cual los registros de zona pueden ser actualizados de una o más fuentes externas. La limitación fundamental de esta especificación es que un nuevo dominio o zona no puede ser añadido o eliminado de forma dinámica. Todos los registros dentro de una zona existente se pueden agregar, cambiar o eliminar con la excepción del RR SOA que no se puede añadir o eliminar porque esencialmente sería añadir o eliminar la zona.

DDNS se describe normalmente en combinación con características de DNS seguro −en concreto, TSIG RFC 2845 [43] y TKEY RFC 2930 [53]− DDNS, sin embargo, no exige ni se basa en las características TSIG/TKEY. La razón por la cual las dos funciones están estrechamente unidas es que al permitir DNS dinámico, los archivos de zona pueden estar abiertos a la posibilidad de la corrupción o el envenenamiento de fuentes maliciosas. La arquitectura del sistema puede además eliminar el riesgo mediante el posicionamiento bajo parámetros seguros tanto del servidor de nombres de destino como de los hosts que está autorizado para actualizar. El poder real, sin embargo, de DDNS es que los usuarios remotos y distribuidos son capaces de actualizar y controlar de forma semiautónoma las configuraciones del dominio. Bajo estas circunstancias es recomendable para los usuarios de DNS dinámico utilizar siempre los procedimientos TSIG/TKEY para autenticar las peticiones entrantes.

6. Anexo F

6.1 - Configuración de BIND

En este anexo introduciremos definiciones y nomenclaturas propias del programa BIND. Estas se utilizan para describir ejemplos de tipos de servidores configurados en el BIND. También incluiremos aquí las configuraciones realizadas durante las pruebas en la maqueta. Más información acerca de las funcionalidades que BIND se puede encontrar en su manual [54].

El archivo de configuración de BIND se llama named.conf y su localización depende del sistema operativo utilizado. En Ubuntu se encuentra en /etc/named/, donde también se deben localizar los archivos de zona que el servidor utiliza. Utilizando un conjunto de cláusulas y sentencias, se pueden obtener todas las posibles funciones del BIND.

Se utiliza el término *cláusula* para describir un grupo declaraciones relacionadas que pueden aparecer en el archivo named.conf. El formato completo y el diseño del archivo named.conf se describe más adelante, pero lo siguiente identifica algunas cláusulas importantes y declaraciones utilizadas en el archivo de configuración:

- La declaración de grupos *Options Clause,* Cláusula de Opciones, controla el comportamiento global del servidor de nombres. En algunos casos, las declaraciones globales pueden ser anuladas en cláusulas específicas, como las cláusulas de la zona.
- La declaración de grupos *Zone Clause,* Cláusula de Zona, refieren a zonas específicas dentro de la configuración, las declaraciones en este tipo de cláusula pueden sobrescribir sentencias de en cláusulas más generales.
- El *Type Statement* es utilizado dentro de una cláusula de zona y define como el servidor de nombres actuará para la zona específica −por ejemplo, puede actuar como maestro o como esclavo para la zona−.
- *Recursion Statement* controla si las consultas recursivas son soportadas o no. El almacenamiento en cache –caching− es un forma de recursión, por lo tanto, esta declaración controla el suministro del servicios caching en el servidor de

nombres. Esta declaración puede aparecer en una cláusula global de opciones o una cláusula de consulta –view clause–. Por defecto, BIND 9 admite las consultas recursivas y por lo tanto, ofrece caching.

- El *file statement* se utiliza para dentro de una cláusula de zona para definir el archivo donde se encuentra almacenado el archivo de zona

6.1.1 - Configuración de un Servidor Autoritativo

Un servidor de nombres autoritativo debe poseer el archivo de zona completo. En nuestro proyecto trabajamos únicamente con servidores autoritativos maestro, por lo que dejaremos la configuración de un servidor esclavo como lectura complementaria. La configuración de una zona se hace dentro de una cláusula de zona como se muestra a continuación:

```
zone "edu.uy." {
       type master;
       file "/etc/bind/zona-uy";
};
```

Aquí se define el tipo de servidor, maestro, y donde se encuentra el archivo de zona con los registros de recursos. En caso de estar sirviendo una zona firmada la sentencia *file* debe apuntar al archivo de zona firmado, el cual en BIND se identifica con la extensión .signed. El archivo named.conf debe incluir una cláusula de zona por cada zona por la cual este sea autoritativo.

Para definir el tipo de servicios que el servidor realizará se utiliza una cláusula de opciones. En estas se especifican las características generales del servidor de nombres.

```
options{

       recursion no;
       DNSSec-validation yes;
       DNSSec-enable yes;

};

zone "." {
       type master;
       file "/etc/bind/zona-uy";
};
```

Un servidor de nombre solo autoritativo debe incluir una sentencia de no-recursión, la cual imposibilita al servidor a ofrecer servicio de consultas recursiva a cualquier cliente. Las sentencias DNSSec-validation y DNSSec-enable configuran al servidor a realizar todas las validaciones y operaciones DNSSec que sean pertinentes. En el caso de un servidor autoritativo que utiliza NSEC para brindar autenticidad de no existencia, no se realizará ningún tipo de operación DNSSec aparte de enviar más registros. En caso de estar usando NSEC3, sí se realizarán operaciones criptográficas como se vio en la sección 6.3 -

6.1.2 - Configuración de un servidor recursivo

La configuración de un servidor recursivo utilizada en nuestras pruebas se limitó a establecer la zona *hint* y a configurar el ancla de confianza en caso de estar operando DNSSec.

```
options{

       recursion yes;
       DNSSec-validation yes;
       DNSSec-enable yes;

};

zone “.” {
       type hint;
       file “/etc/bind/root-servers”;
};
```

En este caso, la configuración *recursion yes* permite la resolución de consultas de forma recursiva para cualquier cliente. Configuraciones más avanzadas permiten limitar el acceso a este servicio en función de la red o IP de origen. Todo servidor recursivo debe estar configurado con la zona raíz, definida bajo el tipo *hint,* esto le dice al servidor que conoce las direcciones IP de la zona raíz que puede utilizar sus datos para comenzar una búsqueda recursiva. El archivo root-servers contiene una lista de los servidores de nombres autoritativos de la zona raíz.

En caso de estar trabajando con zonas firmadas, el servidor recursivo debe contener un ancla de confianza de la cual poder construir una cadena de confianza. Existen dos maneras de realizar esto, utilizaremos la más sencilla.

```
trusted-keys {
  "." 257 3 8
    "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjF
     FVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh/RStIoO8g0NfnfL2MTJRkxoX
     bfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/Efucp2gaD
     X6RS6CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3LQpz
     W5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGOYl7OyQdXfZ57relS
     Qageu+ipAdTTJ25AsRTAoub8ONGcLmqrAmRLKBP1dfwhYB4N7knNnulq
     QxA+Uk1ihz0=";
};
```

Por medio de esta sentencia, se define una clave pública KSK de la raíz, la cual se establece como una clave de confianza y se puede utilizar para validar datos.

Una práctica común en la administración de un servidor recursivo es limitar el acceso a los servicios de recursividad. En BIND esto se puede realizar en la cláusula de opciones de la siguiente manera:

```
allow-recursion { redes; };
```

En *redes* se indican las redes IPv4 que tienen permitido acceder a los servicios de recursividad ofrecidos por el servidor recursivo.

6.1.3 - Name Server Control Utility –rndc–

La aplicación *rndc* incluida en el paquete de BIND, se utiliza para controlar el servidor de nombres named en tiempo de ejecución. Cualquier tipo de cambios en el archivo de configuración o en las zonas de autoridad del servidor, no se aplicarán a la ejecución del programa a menos que el named sea reiniciado, provocando una interrupción en los servicios ofrecidos por el servidor. Para evitar esto, rndc ofrece varias funcionalidades que permiten actualizar, modificar, etc. durante la operación del servidor sin interrumpirlos. Los comandos utilizados durante el proyecto se comentan a continuación junto con una breve explicación, más información se puede encontrar en [54] y [45].

- rndc reload: Recarga el archivo de configuración, named.conf, y las zonas configuradas en el mismo.
- rndc flush: Borra el cache del servidor recursivo.
- rndc dumpdb: Guarda el cache en un archivo de texto.
- service bind9 restart: reinicia el servidor de nombre interrumpiendo su funcionamiento. Este comando es necesario para que named actualice cambios en el sistema que rndc no puede hacer, como cambios en la configuración de la interfaz de red.

6.1.4 - Firma y Roll-Over Automáticos

BIND permite la automatización tanto del procedimiento de firma de la zona como los roll-over de las claves. BIND mantiene un registro de todas las claves DNSSec disponibles en un directorio local. De estas claves, BIND extrae la información temporal de las claves –período de validez– y genera un calendario de roll-over en función de esas fechas. A medida que las claves van caducando BIND inicia de forma automática los roll-over correspondientes. De la misma forma, a medida que las firmas digitales se vencen, el programa re-firma los registros paulatinamente a medida que se precisan.

Dentro de la clausula de opciones se debe indicar que el servidor operará de forma automática insertando esta sentencia:

```
auto-DNSSec maintain;
```

Un roll-over que tiene mayor importancia y debe realizarse de forma automática es el de las anclas de confianza almacenadas en los resolvers validadores. En la sección 7.4 - se especifican el mecanismo que se utiliza en la actualización de este tipo de claves. Para indicar en BIND que se desea actualizar esta clave de manera automática, se debe especificar la calve de confianza inicial, a partir de la cual se validarán las próximas claves de confianza. Es importante aclarar que DNSSec no incorpora mecanismos in-band para la autenticar la primer clave de confianza que se configura en el resolver. Sin embargo, estas vienen ya configuradas en los servidores de nombre como BIND, o se pueden bajar de páginas seguras de Internet bien conocidas.

Para declarar en BIND esta clave de confianza inicial se usa la sentencia managed-keys, la cual se utiliza para definir la primera ancla de confianza de la raíz –o de cualquier otra isla de seguridad–. Las claves sucesivas serán autenticadas por esta calve, y podrán ser luego utilizadas también como anclas de confianza.

```
managed-keys {
  "." initial-key 257 3 8
    "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjF
     FVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh/RStIoO8g0NfnfL2MTJRkxoX
     bfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/Efucp2gaD
     X6RS6CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3LQpz
     W5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGOYl7OyQdXfZ57relS
     Qageu+ipAdTTJ25AsRTAoub8ONGcLmqrAmRLKBP1dfwhYB4N7knNnulq
     QxA+Uk1ihz0=";
};
```

6.1.5 - Transferencia de zona

Un maestro obtiene los datos de la zona a partir de un archivo de zona almacenado localmente, a diferencia del esclavo que obtiene sus datos a través de una operación de transferencia de zona desde el maestro. Este punto aparentemente trivial implica que, es posible tener cualquier cantidad de maestros para cualquier zona si eso tiene sentido operativo. Los cambios en los archivos de zona deben sincronizarse entre los maestros de forma manual o automático.

Un servidor de nombres maestro puede anunciar –mediante mensajes NOTIFY– cambios de zona a los servidores esclavos. Esto asegura que los cambios de zona se propaguen rápidamente a los esclavos, en lugar de simplemente esperar a que el esclavo encueste por cambios en el RR SOA. BIND por defecto automáticamente notificará a todos los servidores de nombres definidos en los registros NS para la zona.

Los mensajes NOTIFY se pueden desactivar mediante el uso declarado de la configuración "*notify no*" en el archivo named.conf de BIND, en la cláusula de zona para dominio.

Cuando un servidor DNS que es maestro para una o más zonas recibe una consulta de una zona para la cual no es maestro o un esclavo, éste actuará como se ha configurado. En BIND 9, este comportamiento se define en el archivo named.conf de la siguiente manera:

- Si el comportamiento caching está permitido y se permiten consultas recursivas, el servidor responderá por completo la solicitud o devolverá un error.
- Si el comportamiento caching está permitido y solo se permiten consultas iterativas –no recursivas–, el servidor puede responder con la respuesta completa si está en el caché, una referencia –almacenada en un archivo local– o devolver un error.
- Si el comportamiento caching no está permitido –un servidor solo autoritativo–, el servidor devolverá una referencia o un error.

El acto de transferencia de zona, puede ser visto como una delegación de autoridad sobre la misma zona hacia esclavos durante un período de tiempo definido en el registro SOA dado por el valor *expiry*. Por lo tanto permite que el esclavo responda con autoridad a las consultas durante ese periodo de tiempo. Momento en el cual deberá solicitar otra transferencia desde el maestro.

Un servidor esclavo intenta actualizar los registros de la zona cuando el parámetro *refresh* de los RR SOA es alcanzado. Si un fallo ocurre, el esclavo periódicamente tratará de alcanzar algún maestro cada retry segundos. Si un esclavo todavía no ha alcanzado al maestro DNS cuando el tiempo de expiración –expiry– del RR SOA de la zona sea alcanzado, éste dejará de responder preguntas para la zona. El esclavo no usará datos que hayan expirado.

7. Anexo G

7.1 - Datos de la prueba del servidor autoritativo sin firmar

Pps	Pps Configurado	Mbps	Pkt Totales	Pkt Enviados	Queries	Respuestas	IDLE Sin Firmar	PIDStat - NAMED Sin Firmar	Ratio: R/Q
195	200	0,325	10000	5000	5000	5000	79,47	8,4	1
238	400	0,544	20000	10000	10000	10000	70,62	16,74	1,00
565	600	0,945	40000	20000	20000	20000	61,03	23,61	1,00
741	800	1,214	44000	22000	22000	22000	54,12	31,85	1,00
911	1000	1,523	56000	28000	28000	28000	48,11	36,71	1,00
1275	1440	2,129	80000	40000	40000	40000	37,48	47,17	1,00
1464	1680	2,446	92000	46000	46000	46000	30,84	52,63	1,00
1813	2140	3,027	111913	56000	55995	55916	20,13	65,18	1,00
2000	2380	3,323	123438	62000	61850	61586	16,55	70,71	1,00
2148	2630	3,581	137630	69000	68974	68654	7,83	79,61	1,00
2325	2900	3,839	152613	77000	76938	75671	5,6	82,78	0,98
2516	3170	4,17	169009	85000	84972	84033	3,28	85,52	0,99
2729	3500	4,503	185785	94000	93923	91858	1,77	86,91	0,98
2843	3700	4,641	192654	98000	97766	94885	1,2	86,86	0,97
2986	3900	4,454	200980	102000	101981	98995	0,73	88,56	0,97
3127	4200	4,861	206463	106000	105973	100490	0,76	87,95	0,95
3283	4400	4,687	212647	110000	109828	102815	0,55	87,56	0,94
3445	4700	5,538	223517	115000	114952	108565	0,36	87,57	0,94
3741	5100	5,96	230733	120000	119762	110969	0,05	88,95	0,93
3878	5400	6,138	239995	125000	124869	115126	0,08	87,83	0,92
4146	5700	6,361	244945	130000	129815	115128	0,09	89,01	0,89
4362	6100	6,524	248394	135000	134730	113664	0,04	87,78	0,84
4483	6500	6,254	261992	145000	144849	117139	0,05	89,45	0,81
4873	7000	6,741	269224	155000	154713	114511	0	87,96	0,74
5081	7500	6,911	298398	175000	174825	123571	0	88,51	0,71

5342	8000	6,95	290631	175000	174914	115717	0	86,74	0,66
5540	8500	7,097	302662	185000	184647	118013	0	88,27	0,64
5625	8500	7,129	301512	185000	184735	116777	0,04	87,26	0,63
5856	9000	7,214	311638	195000	194873	116765	0,04	87,3	0,60
6219	10000	7,745	329893	205000	204341	125550	0	86,4	0,61
7804	15000	8,446	326124	225000	224547	101575	0	85,5	0,45
9155	20000	9,101	337642	245000	244624	93016	0	85,67	0,38

7.2 - Datos de la prueba del servidor autoritativo firmado

Pps	Pps Configurado	Mbps	Pkt Totales	Queries	Respuestas	IDLE Firmado	PIDStat -NAMED Firmado	Ratio: R/Q
195	200	0,75	20000	10000	10000	77,95	9,25	1,00
383	400	1,49	24000	12000	12000	67,79	18,21	1,00
566	600	2,19	30000	15000	15000	58,42	26,4	1,00
743	800	2,88	40000	20000	20000	51,29	32,87	1,00
914	1000	3,48	60000	30000	30000	41,54	40,94	1,00
1287	1440	4,97	69972	35000	34972	31,42	51,1	1,00
1464	1680	5,66	89895	44989	44906	25,17	58,22	1,00
1820	2140	7,02	123802	62000	61802	12,08	74,54	1,00
1979	2380	7,62	137546	68963	68583	6,92	78,97	0,99
2146	2630	8,24	153322	76981	76341	3,41	83,83	0,99
2342	2900	8,84	167507	84954	82553	2,73	85,04	0,97
2515	3170	9,52	185721	93996	91725	1,27	86,6	0,98
2748	3500	10,10	190190	97954	92236	0,41	87,11	0,94
2861	3700	10,61	198851	101866	96985	0,68	87,05	0,95
3001	3900	11,07	205998	105812	100186	0,32	86,4	0,95
3170	4200	11,52	211912	109887	102025	0,14	86,66	0,93
3283	4400	12,01	222285	115000	107285	0,09	87,14	0,93
3474	4700	12,20	229337	119781	109556	0,09	87,52	0,91
3729	5100	12,58	230852	124869	105983	0	86,96	0,85
3661	5400	12,63	233703	124982	108721	0	86,95	0,87
4110	5700	12,72	237443	134851	102592	0	87,56	0,76
4363	6100	12,86	248246	144883	103363	0	87,48	0,71
4554	6500	13,03	261261	154982	106279	0	87,11	0,69
4967	7000	13,17	283375	174678	108697	0,14	86,6	0,62
5194	7500	13,23	277876	174935	102941	0	87	0,59
5449	8000	13,38	288240	184894	103346	0	85,76	0,56
5528	8500	13,38	286249	184654	101595	0	86,41	0,55
5784	9000	13,52	311903	204825	107078	0	85	0,52
6246	10000	13,01	331973	224694	107279	0	86,06	0,48
8043	15000	14,45	331079	244589	86490	0	86,06	0,35
10398	20000	15,37	311780	248832	62948	0	85,91	0,25

7.3 - Datos de la prueba del servidor recursivo sin cache sin firmar

Pps	Pps Configurado	Mbps	Pkt Totales	Pkt Enviados	Queries	Respuestas	IDLE Sin Firmar	NAMED Sin Firmar	Ratio: R/Q
10	10	0,053	1000	500	500	500	85	2	1,00
25	25	0,1265	2000	1000	1000	1000	79,46	7,54	1,00
49	50	0,249	6000	3000	3000	3000	71,47	15,59	1,00
74	75	0,3705	8000	4000	4000	4000	71,17	22,91	1,00
98	100	0,4905	10000	5000	5000	5000	54,84	30,2	1,00
123	125	0,6105	10000	5000	5000	5000	44,27	37,26	1,00
147	150	0,7285	20000	10000	10000	10000	39,48	44,23	1,00
171	175	0,8435	20000	10000	10000	10000	34,27	49,92	1,00
195	200	0,958	20000	10000	10000	10000	25,85	56,12	1,00
220	225	1,072	20000	10000	10000	10000	19,64	62,24	1,00
243	250	1,1685	19996	10000	9998	9998	14,23	68,03	1,00
267	275	1,281	19998	10000	9999	9999	8,23	73,95	1,00
290	300	1,316	19988	10000	9995	9993	5,86	76,45	1,00
313	325	1,417	19960	10000	10000	9960	3,5	79,31	1,00
337	350	1,487	19928	10000	9990	9938	0,91	83,64	0,99
360	375	1,517	19594	10000	9988	9606	0	84,02	0,96
383	400	1,606	29128	15000	14960	14168	0	83,91	0,95
406	425	1,6515	28749	15000	14998	13751	0	84,76	0,92
430	450	1,641	27759	15000	14984	12775	0	83,47	0,85
475	500	1,653	26378	15000	14953	11425	0	82,26	0,76
521	550	1,697	25561	15000	15000	10561	0,73	84,14	0,70
565	600	1,6925	24490	15000	14974	9516	0	84,01	0,64
609	650	1,6935	23660	15000	15000	8660	0	83,86	0,58
654	700	1,7145	23032	15000	15000	8032	0	84,26	0,54
698	750	1,742	29979	20000	19976	10003	0	83,24	0,50
741	800	1,772	29469	20000	20000	9469	0	84,31	0,47
784	850	1,8305	36451	25000	25000	11451	0	84,19	0,46
825	900	1,861	35879	25000	25000	10879	0	84,11	0,44
869	950	1,8785	35301	25000	25000	10301	0	84,41	0,41
910	1000	1,885	34642	25000	24879	9763	0	83,86	0,39
948	1050	1,8715	34127	25000	25000	9127	0	83,85	0,37
1035	1150	1,972	47030	35000	34981	12049	0	84,06	0,34
1116	1250	1,9975	45923	35000	34880	11043	0	82,72	0,32
1198	1350	2,0515	45230	35000	35000	10230	0	83,61	0,29
1277	1450	2,1035	50928	40000	39898	11030	0	82,87	0,28
1346	1550	2,1355	50265	40000	39851	10414	0	84,04	0,26
1424	1650	2,1635	49674	40000	40000	9674	0	83,95	0,24

7.4 - Datos de la prueba del servidor recursivo sin cache firmado

Pps	Pps Configurado	Mbps	Pkt Totales	Pkt Enviados	Queries	Respuestas	IDLE Firmado	NAMED Firmado	Ratio: R/Q
10	10	0,3175	2000	1000	1000	1000	59	27,6	1,00
15	15	0,468	1000	500	500	500	44,83	41,52	1,00
20	20	0,612	1000	500	500	500	31,17	55,33	1,00
25	25	0,7475	2000	1000	1000	1000	20,76	65	1,00
30	30	0,851	2000	1000	1000	1000	10,21	74,76	1,00
35	35	0,94	2000	1000	1000	1000	2,13	84,78	1,00
40	40	0,947	2000	1000	1000	1000	1,52	84,71	1,00
49	50	1,0245	5000	2500	2500	2500	0,22	86,66	1,00
98	100	1,0185	8000	4000	4000	4000	0	86,27	1,00
123	125	1,014	8469	4500	4500	3969	0	87,15	0,88
147	150	1,0335	8655	5000	5000	3655	0	86,74	0,73
171	175	1,066	13127	8000	8000	5127	0	87,16	0,64
220	225	1,0935	15022	10000	9999	5023	0	87,2	0,50
243	250	1,094	14494	10000	10000	4494	0	86,62	0,45
266	275	1,0935	14034	10000	10000	4034	0	86,16	0,40
289	300	1,1125	20680	15000	14999	5681	0	87,01	0,38
313	325	1,142	20256	15000	15000	5256	0	87	0,35
336	350	1,1505	19832	15000	15000	4832	0	86,46	0,32
359	375	1,158	19441	10000	15000	4441	0	86,95	0,30
381	400	1,156	19182	15000	14990	4192	0	86,25	0,28
405	425	1,172	18939	15000	14989	3950	0	86,64	0,26
428	450	1,1795	18717	15000	14998	3719	0	86,91	0,25
474	500	1,1775	18261	15000	14999	3262	0	85,36	0,22
518	550	1,1485	17975	15000	14995	2980	0	85,45	0,20
562	600	1,216	17705	15000	15000	2705	0	86,19	0,18
607	650	1,242	18655	16000	16000	2655	0	85,46	0,17
652	700	1,2495	18449	16000	16000	2449	0	85,56	0,15
694	750	1,3	22939	20000	20000	2939	0	86,24	0,15
736	800	1,316	22746	20000	20000	2746	0	86,05	0,14

7.5 - Datos de la prueba del servidor recursivo con cache sin firmar

Pps	Pps configurado	Mbs	Pkt Enviados	Queries	Respuestas	CPU IDLE	PIDStat - NAMED	Ratio: R/Q	% Cache
50	50	0,1095	2000	2000	2000	81,16	6,93	1,00	68,6
99	100	0,1805	4000	4000	4000	74,09	13,03	1,00	68,95
147	150	0,3255	6000	6000	6000	67,3	19,99	1,00	69,3

195	200	0,4065	8000	8000	8000	60,65	25,83	1,00	69,15
243	250	0,5385	10000	10000	10000	54,15	32,44	1,00	69,26
290	300	0,643	12000	12000	12000	47,57	37,79	1,00	69,47
338	350	0,7445	14000	14000	14000	41,73	43,35	1,00	69,73
475	500	1,048	20000	20000	20000	28	56,7	1,00	69,75
610	650	1,349	26000	25995	25956	14,16	70,82	1,00	69,61
653	700	1,443	28000	27971	27859	10,09	74,74	1,00	69,71
741	800	1,637	30000	29956	27750	3,18	82,61	0,93	69,71
827	900	1,764	34000	33930	32108	0,24	84,61	0,95	71,17
909	1000	1,8265	38000	37996	32984	0	85,47	0,87	73,62
1074	1200	1,914	44000	43884	32070	0	85,84	0,73	78,04
1243	1400	2,0065	50000	49869	31121	0	85,34	0,62	81,25
1404	1600	1,97	60000	59923	33024	0	85,18	0,55	83,46
1564	1800	2,057	67000	66768	32589	0	85,44	0,49	85,24
1727	2000	2,28	66000	65956	28792	0	85,34	0,44	86,96
1859	2200	2,2215	73000	72796	29423	0	85,6	0,40	87,91
2007	2400	2,4365	76000	75834	28012	0	85,6	0,37	88,89
2123	2600	2,5155	83000	82962	28955	0	85,21	0,35	89,42
2255	2800	2,6015	95000	94849	31315	0	84,67	0,33	90,03469

7.6 - Datos de la prueba del servidor recursivo con cache firmado

Pps	Mbs	Pkt Totales	Pkt Enviados	Queries	Respuestas	CPU IDLE	PIDStat - NAMED	Ratio: R/Q	%cache
50	0,1695	4000	2000	2000	2000	80,86	12,65	1,00	68,6
99	0,322	8000	4000	4000	4000	63,84	24,07	1,00	68,95
148	0,495	12000	6000	6000	6000	63,8	24,24	1,00	69,3
196	0,6495	10000	5000	5000	5000	48,47	38,23	1,00	68,84
290	0,9725	24000	12000	12000	12000	41,78	44,58	1,00	69,47
337	1,2126	18000	9000	9000	9000	33,8	52,2	1,00	69,29
475	1,5815	39968	20000	20000	19968	15,28	70,32	1,00	69,78
609	2,0285	51763	26000	26000	25763	1,57	84,53	0,99	69,81
653	2,1245	54994	28000	28000	26994	0,22	85,85	0,96	70,61
739	2,1565	36930	20000	20000	16930	0	85,54	0,85	74,29
825	2,2035	40379	23000	23000	17379	0	85,91	0,76	77,03
908	2,2375	38576	23000	23000	15576	0	85,32	0,68	79,35
1073	2,336	45505	29000	29000	16505	0	85,42	0,57	82,57
1233	2,3765	44531	30000	30000	14531	0	85,12	0,48	85,48
1386	2,4825	50146	35000	35000	15146	0	85,45	0,43	87,03

7.7 - Datos de la prueba NXDOMAIN usando NSEC

Pps	Pps Config	Mbps	CPU IDLE	PidStat-NAMED	Respuestas	Consutlas	% perdidas	Ratio
50	50	0,424	62,61	24,79	1250	1250	0,00	1,00
99	100	0,8335	39,2	47,42	7480	7500	0,27	1,00
123	125	1,0415	26,14	60,25	3125	3125	0,00	1,00
148	150	1,2445	14,56	71,44	3750	3750	0,00	1,00
172	175	1,458	4,5	82,04	4375	4375	0,00	1,00
195	200	1,6265	0	87,25	4911	5000	1,78	0,98
243	250	1,632	0	86,7	4947	6300	21,48	0,79
290	300	1,677	0	87,16	4981	7500	33,59	0,66
337	350	1,6865	0	85,63	4945	8750	43,49	0,57
383	400	1,7195	0	86,94	4980	10000	50,20	0,50
429	450	1,8285	0	93,56	5268	11250	53,17	0,47
474	500	1,7685	0	86,81	4993	12500	60,06	0,40

7.8 - Datos de la prueba NXDOMAIN usando NSEC3

Pps	Pps Config	Mbps	CPU IDLE	PidStat-NAMED	Respuestas	Consutlas	% perdidas	Ratio
50	50	0,557	38,45	49,02	1250	1250	0,00	1,00
74	75	0,829	13,97	74,44	2000	2000	0,00	1,00
99	100	1,015	0	88,66	2331	2500	6,76	0,93
123	125	1,028	0	88,71	2333	3125	25,34	0,75
147	150	1,044	0	88	2339	3750	37,63	0,62
161	175	1,0465	0	88,75	2314	4375	47,11	0,53
195	200	1,0725	0	88,26	2353	5000	52,94	0,47
243	250	1,097	0	88,61	2375	6300	62,30	0,38
290	300	1,117	0	87,76	2341	7500	68,79	0,31
335	350	1,1455	0	88,95	2353	8750	73,11	0,27
382	400	1,1705	0	88,41	2360	10000	76,40	0,24
428	450	1,1915	0	88,45	2357	11250	79,05	0,21

8. Bibliografía

[1] P. Mockapretis, «RFC 882: Domain Names, Concepts and facilities,» IETF, 1983.

[2] P. Mockapetris, «RFC 883: Domain Names, Implementation and Specification,» IETF, 1983.

[3] P. Mockapetris, «RFC 1034: Domain Names, Concepts and Facilities,» IETF, 1987.

[4] P. Mockapetris, «RFC 1035: Domain Names, Impementation and Specification,» IETF, 1987.

[5] D. Eastlake, «RFC 2606: Reserverd Top Level DNS Names,» IETF, 1999.

[6] ICANN, «ICANN's Major Agreements and Related Reports,» [En línea]. Available: http://www.icann.org/en/about/agreements.

[7] E. Lewis, «RFC 4592: The Role of Wildcards in the Domain Name System,» NeuStar, 2006.

[8] ICANN, «Root Servers,» ICANN, [En línea]. Available: http://www.root-servers.org/. [Último acceso: 13 04 2012].

[9] D. Conrad, «RFC 3225: Indicating Resolver Support of DNSSEC,» Nominum, Inc., 2001.

[10] R. Aitchison, Pro Dns and BIND 10, Apress, 2011.

[11] O. Kolkman y M. Santcroos, «DNS Threat Analysis,» NLnet Labs, Amsterdam, 2007.

[12] S. Bellovin, « Using the Domain Name System for System Break-Ins,» IETF, 1995.

[13] D. Eastlake, «RFC 2065: Domain Name System Security Extensions,» IETF, 1997.

[14] D. Eastlake, «RFC 2535: Domain Name System Security Extensions,» IETF, 1999.

[15] R. Arends, «RFC 4033: DNS Security Introduction and Requirements,» IETF, 2005.

[16] R. Arends, «RFC 4034: Resource Records for the DNS Security Extensions,» IETF, 2005.

[17] R. Arends, «RFC 4035: Protocol Modifications for the DNS Security Extensions,» IETF, 2005.

[18] VeriSignLabs, «DNSSec Scoreboard,» [En línea]. Available: http://scoreboard.verisignlabs.com/. [Último acceso: 19 04 2012].

[19] N. I. o. S. a. Technology, «Special Publication 800-53,» NIST, 2011.

[20] EsEt, «Blog de Laboratorio,» [En línea]. Available: http://blogs.eset-la.com/laboratorio/2011/07/21/envenenamiento-dns-phishing-banco-brasil/. [Último acceso: 10 04 2012].

[21] Wikipedia, «Ley SOPA,» [En línea]. Available: http://es.wikipedia.org/wiki/Stop_Online_Piracy_Act#cite_note-new_flap-73. [Último acceso: 10 04 2012].

[22] IANA, «Root Anchors,» [En línea]. Available: http://data.iana.org/root-anchors/. [Último acceso: 30 04 2012].

[23] M. Andrews, «RFC 2308: Negative Caching of DNS Queries,» CSIRO, 1998.

[24] B. Laurie, G. Sisson y R. Arends, «RFC 5155: DNS Security Hashed Authenticated Denial of Existence,» IETF, 2008.

[25] Wikipedia, «MD5,» [En línea]. Available: http://en.wikipedia.org/wiki/MD5. [Último acceso: 10 04 2012].

[26] Wikipedia, «SHA1,» [En línea]. Available: http://es.wikipedia.org/wiki/Secure_Hash_Algorithm. [Último acceso: 10 04 2012].

[27] O. Kolkman, «RFC 4641: DNSSEC Operational Practices,» NLnet Labs, 2006.

[28] F. Ljunggren, «DNSSEC Practice Statement for the Root Zone KSK Operator,» ICANN, 2010.

[29] I. S. Consortium, «ISC,» [En línea]. Available: http://www.isc.org/. [Último acceso: 15 04 2012].

[30] FIPS, «SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC,» NIST, 2001.

[31] J. Jansen, «DNSSEC Key Maintenance Analysis,» NLnet Labs, 2008.

[32] M. StJohns, «RFC 5011: Automated Updates of DNS Security Trust Anchors,» IETF, 2007.

[33] W. Fundation, «Wireshark,» [En línea]. Available: http://www.wireshark.org/. [Último acceso: 15 04 2012].

[34] A. Turner, «Tcpreplay,» [En línea]. Available: http://tcpreplay.synfin.net/. [Último acceso: 15 04 2012].

[35] O. Project, «OpenDNSSec,» [En línea]. Available: http://www.opendnssec.org/. [Último acceso: 15 04 2012].

[36] N. Labs, «NSD,» [En línea]. Available: http://www.nlnetlabs.nl/projects/nsd/. [Último acceso: 15 04 2012].

[37] N. Labs, «Unbound,» [En línea]. Available: http://unbound.net/. [Último acceso: 15 04 2012].

[38] S. Godard, «Sysstat Home Page,» [En línea]. Available: http://sebastien.godard.pagesperso-orange.fr/. [Último acceso: 15 04 2012].

[39] Google, «Google Official blog,» [En línea]. Available: http://googleblog.blogspot.com/2012/02/google-public-dns-70-billion-requests.html. [Último acceso: 5 4 2012].

[40] I. A. Valdés, «Medida de performance y estadisticas de servidores recursivod de DNS,» 2010.

[41] Y. Schaeffer, «NSEC3 Hash Performance,» NLnet Labs, 2010.

[42] dk379, «NSEC3PARAM Salt Hunt,» [En línea]. Available: http://blog.kohmanyuk.me/nsec3param-salt-hunt. [Último acceso: 4 April 2011].

[43] R. Chandramouli, «Secure Domain Name System Deployment Guide SP800-81,» NIST, EEUU, 2006.

[44] ENISA, «Deploying DNSSEC,» 2010.

[45] O. Kolkman, «DNSSec Project Outlline,» NLnet Labs, 2006.

[46] P. Vixie, «RFC 1996: A Mechanism for Prompt Notification of Zone Changes DNS NOTIFY,» IETF, 1996.

[47] P. Vixie, «RFC 2845: Secret Key Transaction Authentication for DNS (TSIG),» ISC, 2000.

[48] SPARTA, «DNSSEC Operations: Setting the Parameters,» SPARTA, 2009.

[49] D. Barr, «RFC 1912: Common DNS Operational and Configuration Errors,» The Pennsylvania State University, Pennsylvania, 1996.

[50] S. Thomson, «RFC 3596: DNS Extensions to Support IP Version 6,» AFNIC, 2003.

[51] D. Lawrence, «RFC 3425: Obsoleting IQUERY,» Nominum, 2002.

[52] G. Huston, «RFC3172: Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa"),» IAB, 2001.

[53] K. Hubbard, «RFC 2050: INTERNET REGISTRY IP ALLOCATION GUIDELINES,» InterNIC, 1996.

[54] M. Ohta, «RFC 1995: Incremental Zone Transfer in DNS,» Tokyo Institute of Technology, 1996.

[55] P. Vixie, «RFC 2136: Dynamic Updates in the Domain Name System (DNS UPDATE),» ISC, 1997.

[56] D. Eastlake, «RFC 2930: Secret Key Establishment for DNS (TKEY RR),» Motorola, 2000.

[57] *BIND 9 Administrator Reference Manual,* 2011.

[58] H.-J. Kim, «Vulnerability Modeling and Simulation,» Korea Information Security Agency.

[59] S. Castro, «Understanding and preparing for DNS evolution,» CAIDA, University of California, San Diego.

[60] J. Trostle, Techniques for improving the security, Springer, 2005.

[61] S. Rose, «NIST DNSSEC workshop notes,» June 2001.

[62] 2nd Global Symposium on DNS Security, Stability and Resiliency, «Measuring the Health of the Domain Name System,» Kyoto University, Kyoto, 2010.

[63] J. Jiang, «Ghost Domain Names: Revoked Yet Still Resolvable,» Tsinghua University.

[64] G. Kambourakis, «Detecting DNS Amplification Attacks,» University of the Aegean, Greece.

[65] H. M. Sun, «DepenDNS: Dependable Mechanism against,» National Tsing Hua University, Taiwan.

[66] G. Guette, «Algorithm for DNSSEC Trusted Key Rollover,» IRISA,, FRANCE.

[67] E. Amberg, «Administración DnsSec,» *LINUX-MAGAZINE,* pp. 58-65.

[68] J. Bau, «A Security Evaluation of DNSSEC with NSEC3,» Stanford University.

[69] S. Thomson, «RFC 1886: DNS Extensions to support IP version 6,» INRIA, 1995.

[70] M. Gieben, «Authenticated Denial of Existence in the DNS,» SIDN, 2012.